Jak tworzymy algorytmy?

Wersja: 5

Wojciech Myszka

2017-12-02 21:39:27 +0100

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...?*

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...?*
4. **Maszyna Turinga?**

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...?*
4. **Maszyna Turinga?**
5. **Pseudojęzyk** — rodzaj formalnego zapisu podobny do...

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...?*
4. **Maszyna Turinga?**
5. **Pseudojęzyk** — rodzaj formalnego zapisu podobny do...
6. **Język programowania.** *Następny semestr!*

Zapis algorytmu

Schemat blokowy

Schemat blokowy (ang. *block diagram, flowchart*) — diagram, na którym procedura, system albo program komputerowy, są reprezentowane przez opisane figury geometryczne połączone liniami zgodnie z kolejnością wykonywania czynności wynikających z przyjętego algorytmu rozwiązania zadania.

Schemat blokowy pozwala dostrzec istotne etapy algorytm i logiczne zależności między nimi.

Zależnie od przedstawianego zagadnienia stosowane są różne zestawy figur geometrycznych zwanych blokami, których kształty reprezentują umownie rodzaje elementów składowych.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.
6. Porównaj ją z tą stojącą przy drzwiach — czy jest większa?

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.
6. Porównaj ją z tą stojącą przy drzwiach — czy jest większa?
7. Jeżeli nie — przejdź do 2

Wybór osoby najwyższej na sali

Wersja słowna

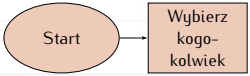
1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.
6. Porównaj ją z tą stojącą przy drzwiach — czy jest większa?
7. Jeżeli nie — przejdź do 2
8. Jeżeli tak — zamienia osobę stojącą przy drzwiach; przejdź do 2

Wybór osoby najwyższej na sali

Schemat blokowy

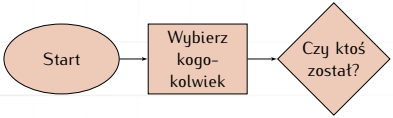
Wybór osoby najwyższej na sali

Schemat blokowy



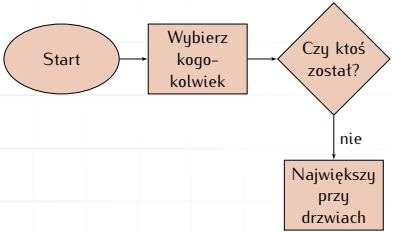
Wybór osoby najwyższej na sali

Schemat blokowy



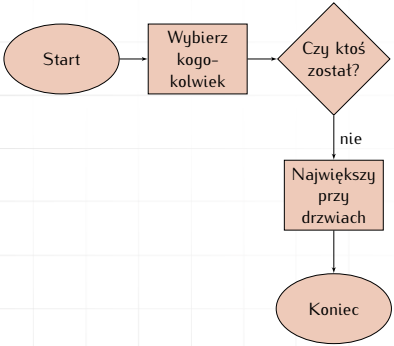
Wybór osoby najwyższej na sali

Schemat blokowy



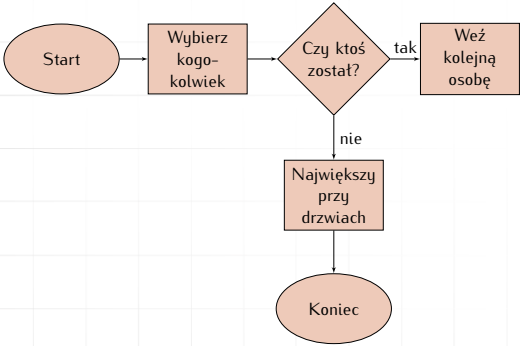
Wybór osoby najwyższej na sali

Schemat blokowy



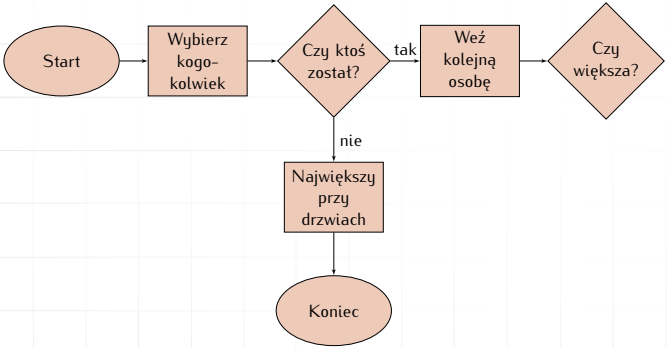
Wybór osoby najwyższej na sali

Schemat blokowy



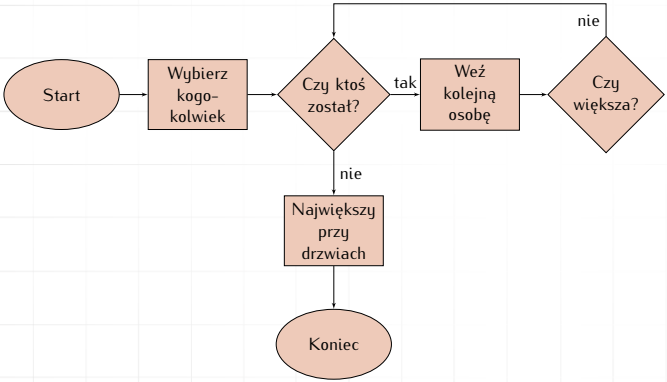
Wybór osoby najwyższej na sali

Schemat blokowy



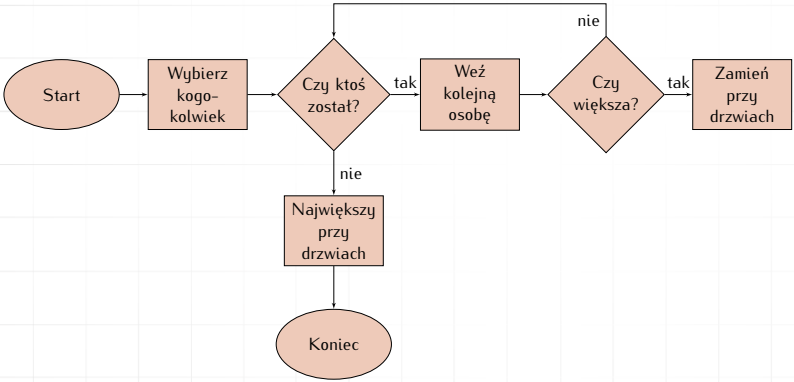
Wybór osoby najwyższej na sali

Schemat blokowy



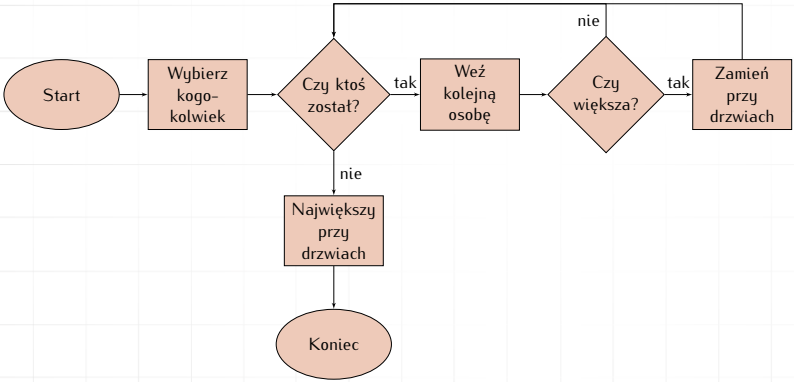
Wybór osoby najwyższej na sali

Schemat blokowy



Wybór osoby najwyższej na sali

Schemat blokowy



Algorytm Euklidesa

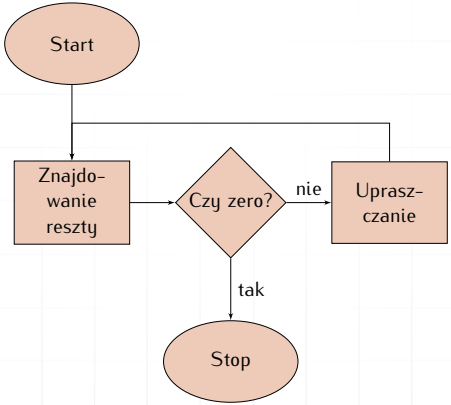
Słownie

1. [Znajdowanie reszty] Podziel m przez n i niech r oznacza resztę z tego dzielenia. (Mamy $0 \leq r < n$.)
2. [Czy wyszło zero?] Jeśli $r = 0$ zakończ algorytm; odpowiedzią jest n.
3. [Upraszczanie] Wykonaj $m \leftarrow n$, $n \leftarrow r$ i wróć do kroku 1.

Algorytm Euklidesa

Schemat blokowy

Zapiszmy teraz Algorytm Euklidesa w postaci schematu blokowego:



Tablice decyzyjne

Tablica decyzyjna to mieszanka warunków i decyzji które należy podejmować w zależności od ich spełnienia.

Tablice decyzyjne

Tablica decyzyjna to mieszanka warunków i decyzji które należy podejmować w zależności od ich spełnienia.

- Alternatywa dla schematu blokowego.
- Bardzo wygodne w przypadku opisu problemów z ogromną ilością decyzji.
- Nieźle nadaje się do opisu problemów „z życia wziętych”.
- Średnie zastosowanie w przypadku problemów obliczeniowych.

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania			
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny		
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj		
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań	Rezygnuj		

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań	Rezygnuj	Rezygnuj	

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań	Rezygnuj	Rezygnuj	Targuj się o dodatki; kup jeśli cena dodatków jest rozsądna

Przykład: sklepik

Kolejny przykład to tablica decyzyjna opisująca działania związane z przyjęciem i realizacją zamówienia.

Tablica uwzględnia też politykę firmy, którą można opisać tak:

1. Firma obsługuje **tylko** zarejestrowanych klientów.
2. Firma dostarcza **tylko** towary znajdujące się na liście towarów.

Sklepik c.d.

C1	Towar na liście	T	N	—	T
C2	Klient zarejestrowany	T	—	N	T
C3	Wystarczający zapas towaru	T	—	—	N
A1	Zarezerwuj towar	X			
A2	Zarejestruj transakcję	X			
A3	Zapisz zamówienie na liście do realizacji				X
A4	Wyślij towar	X			
A5	Odrzuć transakcję		X	X	

Jak się tworzy algorytmy?

Moja odpowiedź jest krótka:

Jak się tworzy algorytmy?

Moja odpowiedź jest krótka:

Nie wiem!

Poszukiwania i wędrówki

Czyli przegląd

1. Bardzo często zachodzi potrzeba **obejścia** wszystkich elementów struktury.
2. W najprostszym przypadku *struktura* zadana jest jawnie (tablica, wektor, drzewo).
3. W wielu przypadkach — trzeba dobrze przyjrzeć się zagadnieniu, żeby strukturę zauważyć.
4. Gdy zadanie ma szereg wariantów — wystarczy przejrzeć je wszystkie

W każdym przypadku działanie algorytmu sprowadza się do przeglądu wszystkich elementów struktury.

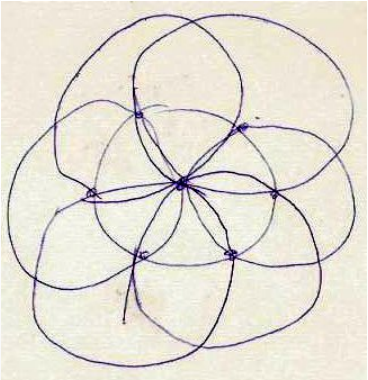
Przegląd

Pętla

1. **Start**
2. $i \leftarrow 0$
3. *Zrób coś...*
4. $i \leftarrow i + 1$
5. **Jeżeli** $i \leq N$ **przejdź do** 3
6. **W przeciwnym razie — Koniec**

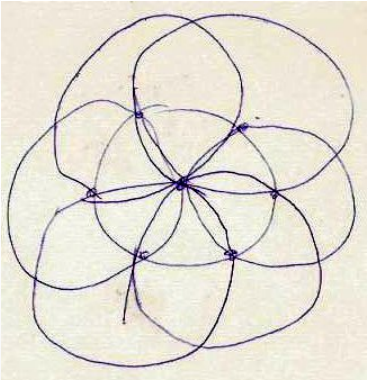
Problem

Ktoś prosił nas o narysowanie „kwiatka” czyli rysunku, wyglądającego jakoś tak:



Problem

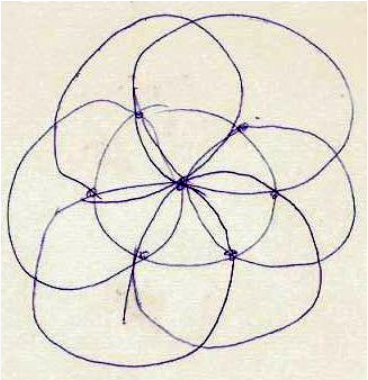
Ktoś prosił nas o narysowanie „kwiatka” czyli rysunku, wyglądającego jakoś tak:



Tylko porządniej...

Problem

Ktoś prosił nas o narysowanie „kwiatka” czyli rysunku, wyglądającego jakoś tak:



Tylko porządniej...
Jak się za to zabrać?

Metody

1. Cyrkiel...

Metody

1. Cyrkiel...
2. Jakiś program graficzny (Corel, OpenOffice.org Draw, cokolwiek...)

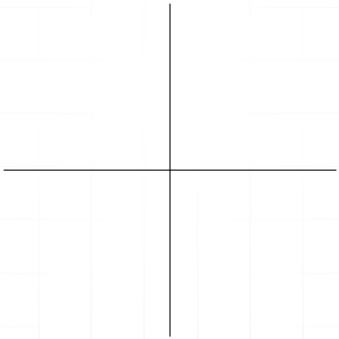
Metody

1. Cyrkiel...
2. Jakiś program graficzny (Corel, OpenOffice.org Draw, cokolwiek...)
3. Skonstruować...

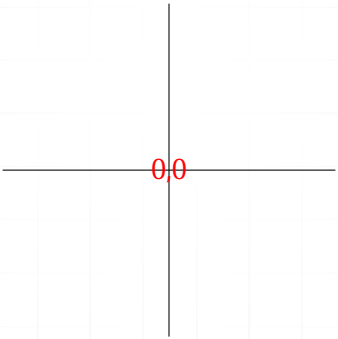
Metody

1. Cyrkiel...
2. Jakiś program graficzny (Corel, OpenOffice.org Draw, cokolwiek...)
3. Skonstruować...
4. Napisać program komputerowy...

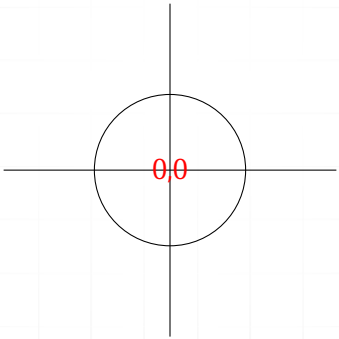
Kwiatek



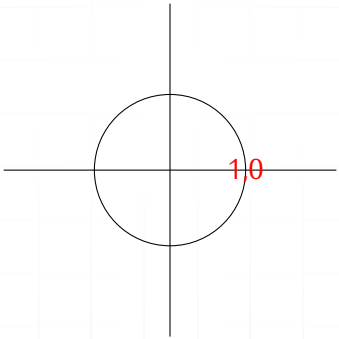
Kwiatek



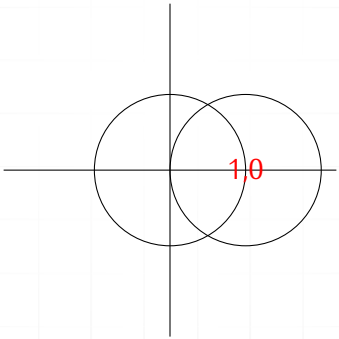
Kwiatek



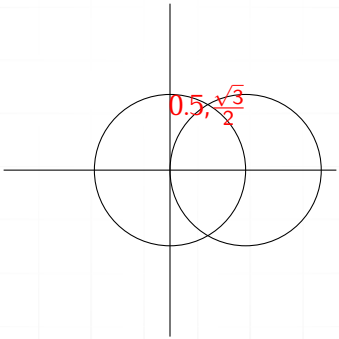
Kwiatek



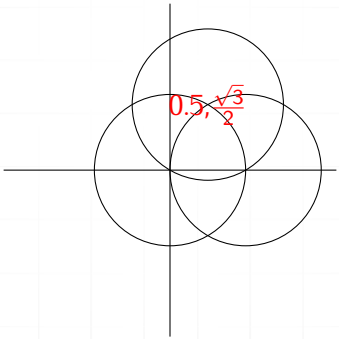
Kwiatek



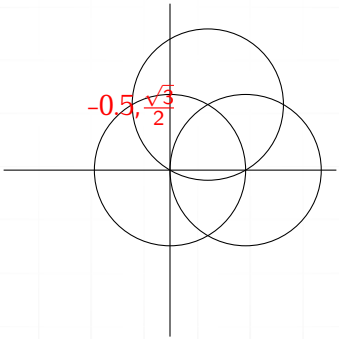
Kwiatek



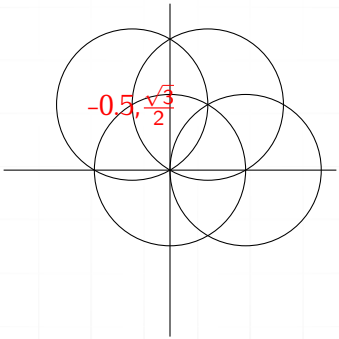
Kwiatek



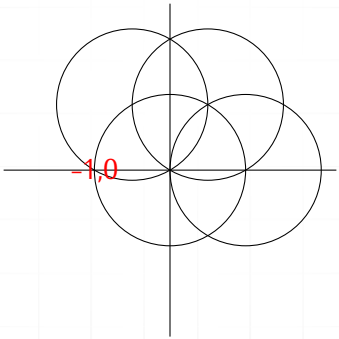
Kwiatek



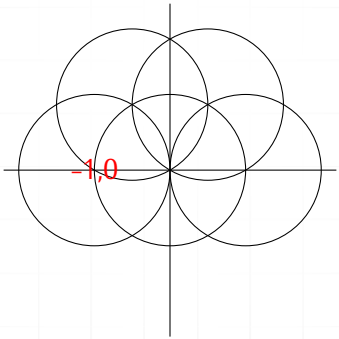
Kwiatek



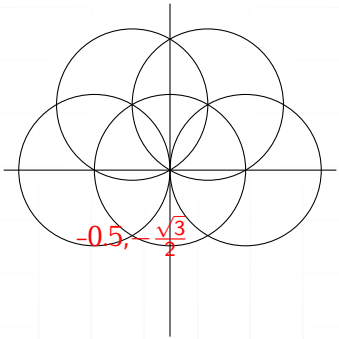
Kwiatek



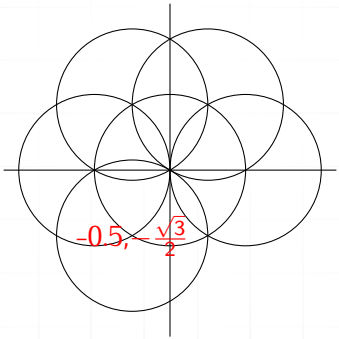
Kwiatek



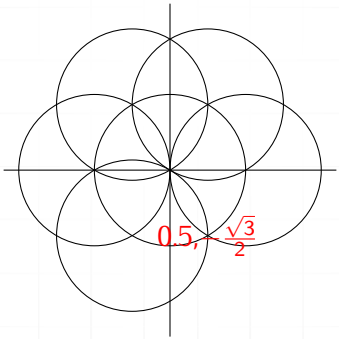
Kwiatek



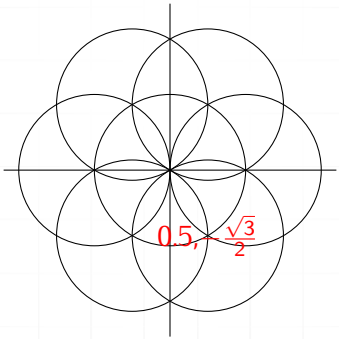
Kwiatek



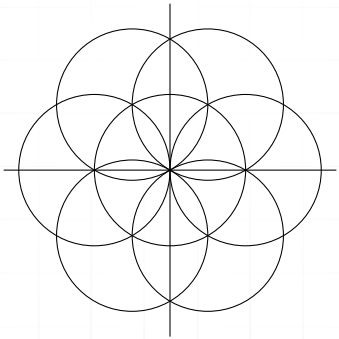
Kwiatek



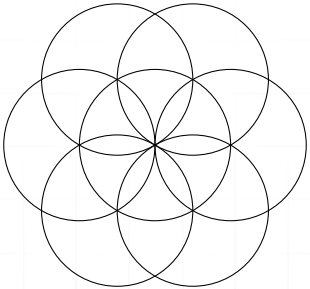
Kwiatek



Kwiatek



Kwiatek



Jak to jest zrobione?

```
\draw (0,0) circle (1cm);
%
\draw (1,0) circle (1cm);
\draw (0.5,0.866) circle (1cm);
\draw (-0.5,0.866) circle (1cm);
\draw (-1,0) circle (1cm);
\draw (-0.5,-0.866) circle (1cm);
\draw (0.5,-0.866) circle (1cm);
```

Kwiatek w pythonie

```
1  #!/usr/bin/env python
2  from turtle import *
3  setup(600,600,300,300)
4  title("Kwiatek")
5  speed(1)
6  up()
7  goto (100,0)
8  down()
9  setheading(90)
10 circle(100)
11 for _ in range(6):
12     setheading(heading() - 60)
13     down()
14     circle(100)
15     setheading(heading() + 60)
16     up()
17     circle(100, 60)
18
19 exitonclick()
```

Definicje

Za wikipedią

Rekursja albo rekurencja (ang. recursion, z łac. recurrere, przybiec z powrotem) to w logice, programowaniu i w matematyce odwoływanie się (np. funkcji lub definicji) do samej siebie.

Przykład

Silnia

Wersja klasyczna

$0! = 1$

$n! = \prod_{i=1}^{n} i$

albo

$n! = 1 \times 2 \times 3 \times \cdots \times n$

Przykład

Silnia

Wersja klasyczna

$0! = 1$

$n! = \prod_{i=1}^{n} i$

albo

$n! = 1 \times 2 \times 3 \times \cdots \times n$

Wersja inna

$0! = 1$

$n! = n \times (n-1)!$

Przykład

Kilka liczb

In[1]:= 10!
Out[1]= 3628800
In[3]:= 20!
Out[3]= 2432902008176640000
In[4]:= 30!
Out[4]= 265252859812191058636308480000000
In[5]:= 40!
Out[5]= 815915283247897734345611269596115894272000000000

Silnia

„Schematy blokowe”

Wersja klasyczna

1. Jeżeli $n = 0$, silnia równa się 1; koniec algorytmu.
2. $silnia \leftarrow 1$
3. Powtarzaj dla i zmieniającego się od 1 do n
   $silnia = silnia \times i$
4. Koniec algorytmu.

Silnia

„Schematy blokowe”

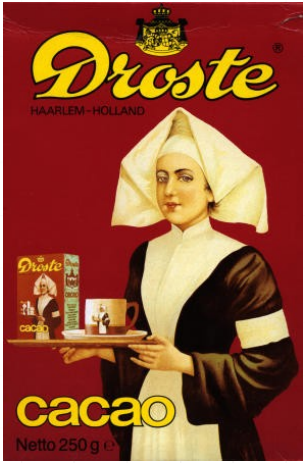
Wersja klasyczna

1. Jeżeli $n = 0$, silnia równa się 1; koniec algorytmu.
2. $silnia \leftarrow 1$
3. Powtarzaj dla i zmieniającego się od 1 do n
   $silnia = silnia \times i$
4. Koniec algorytmu.

Wersja rekurencyjna

1. Funkcja Silnia (parametrem jest n)
2. Jeżeli $n = 0$; *wynik* podstaw 1; koniec programu
3. $wynik = wynik \times$ Silnia$(n - 1)$
4. koniec

Rekurencja na obrazkach: Droste Effect



Rekurencja — Escher

Rekurencja — Escher

Rekurencja — Escher

Rekurencja — Escher

Rekurencja — Escher

Rekurencja — Escher

Rekurencja — Escher

Rekurencja — Escher

Ciąg Fibonacciego

Schemat

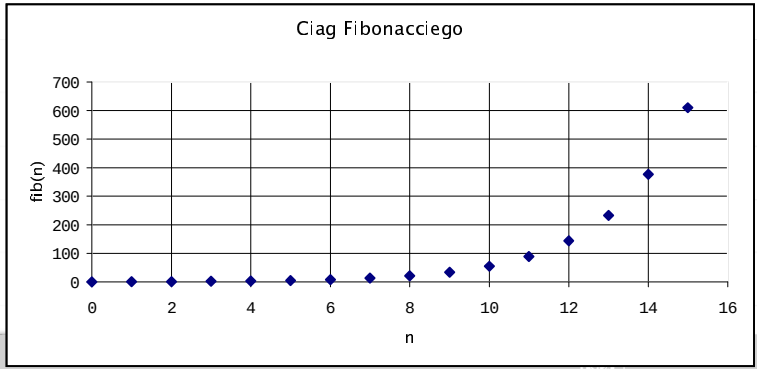
$\text{fib}(0) = 0$
$\text{fib}(1) = 1$
$\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - 2)$, dla $n \geq 2$

Ciąg Fibonacciego

Schemat

fib(0) = 0
fib(1) = 1
fib(n) = fib(n - 1) + fib(n - 2), dla $n \geq 2$

0	0	
1	1	
2	1	=B2+B1
3	2	=B3+B2
4	3	
5	5	
6	8	
7	13	
8	21	
9	34	
10	55	
11	89	
12	144	
13	233	
14	377	
15	610	



Największy wspólny dzielnik

$\gcd(0,n)=n$

$$\gcd(k,n) = \begin{cases} n & \text{dla } k = 0; \\ \gcd(n \bmod k, k) & \text{dla } k > 0. \end{cases}$$

Największy wspólny dzielnik

$\gcd(0,n)=n$

$$\gcd(k,n) = \begin{cases} n & \text{dla } k = 0; \\ \gcd(n \bmod k, k) & \text{dla } k > 0. \end{cases}$$

Porównać ze schematem blokowym („metoda Euklidesa”), który był prezentowany wcześniej

Symbol Newtona

Wersja „normalna”

$$\binom{n}{0} = 1$$

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!}$$

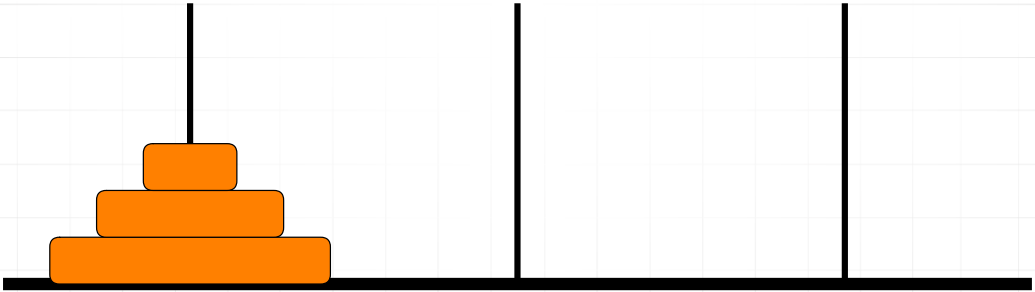
Wersja „rekurencyjna”

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

$$\binom{n}{0} = 1$$

Wieże Hanoi

Prosta zabawka dziecięca — na patyku nanizanych jest pewna liczba krążków tak, że na większym zawsze leży krążek mniejszy. Zadaniem naszym jest umieszczenie wszystkich krążków na sąsiednim „patyku” (korzystając z jednego tylko „patyka” pomocniczego) w tej samej kolejności. Podczas każdego ruchu pamiętać trzeba, że krążek większy nie może znaleźć się nigdy na krążku mniejszym.



Wieże Hanoi

Przykład

Trzy krążki

Wieże Hanoi

Przykład

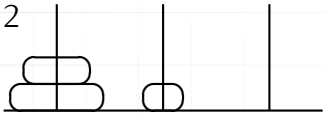
Trzy krążki

1

Wieże Hanoi

Przykład

Trzy krążki

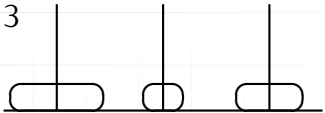


Wieże Hanoi

Przykład

Trzy krążki

3



Wieże Hanoi

Przykład

Trzy krążki

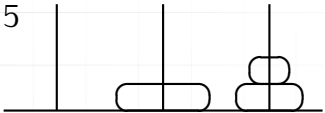
4

Wieże Hanoi

Przykład

Trzy krążki

5

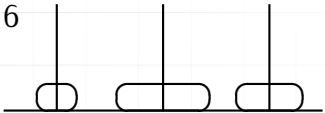


Wieże Hanoi

Przykład

Trzy krążki

6

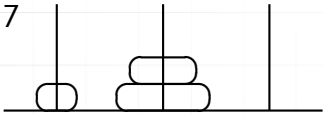


Wieże Hanoi

Przykład

Trzy krążki

7

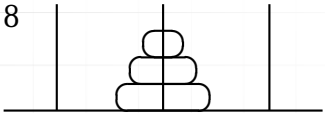


Wieże Hanoi

Przykład

Trzy krążki

8



Wieże Hanoi

Przykład

Dwa krążki

Wieże Hanoi

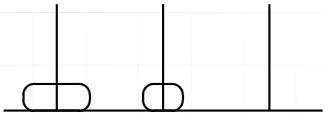
Przykład

Dwa krążki

Wieże Hanoi

Przykład

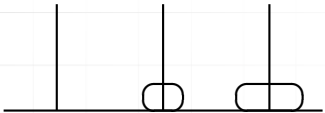
Dwa krążki



Wieże Hanoi

Przykład

Dwa krążki



Wieże Hanoi

Przykład

Dwa krążki

Wieże Hanoi I

Czemu taka nazwa?

„W wielkiej świątyni Benares w Hanoi, pod kopułą, która zaznacza środek świata, znajduje się płytka z brązu, na której umocowane są trzy diamentowe igły, wysokie na łokieć i cienkie jak talia osy.

Na jednej z tych igieł, w momencie stworzenia świata, Bóg umieścił 64 krążki ze szczerego złota. Największy z nich leży na płytce z brązu, a pozostałe jeden na drugim, idąc malejąco od największego do najmniejszego. Jest to wieża Brahma.

Bez przerwy we dnie i w nocy kapłani przekładają krążki z jednej diamentowej igły na drugą, przestrzegając niewzruszonych praw Brahma.

Prawa te chcą, aby kapłan na służbie brał tylko jeden krążek na raz i aby umieszczał go na jednej z igieł w ten sposób, by nigdy nie znalazł się pod nim krążek mniejszy.

Wieże Hanoi II

Czemu taka nazwa?

Wówczas, gdy 64 krążki zostaną przełożone z igły, na której umieścił je Bóg w momencie stworzenia świata, na jedną z dwóch pozostałych igieł, wieża, świątynia, bramini rozsypią się w proch i w jednym oka mgnieniu nastąpi koniec świata".

Kiedy nastąpi koniec świata?

Wieże Hanoi

Koniec świata

- Przy trzech krążkach trzeba 7 ruchów (przy dwu — 3!)

Wieże Hanoi

Koniec świata

- Przy trzech krążkach trzeba 7 ruchów (przy dwu — 3!)
- Gdy krążków jest 64 — trzeba $2^{64} - 1$ ruchów, czyli 18446744073709551615

Wieże Hanoi

Koniec świata

- Przy trzech krążkach trzeba 7 ruchów (przy dwu — 3!)
- Gdy krążków jest 64 — trzeba $2^{64} - 1$ ruchów, czyli 18446744073709551615
- Zakładamy, że przeniesienie krążka zajmuje 1 sekundę.

Wieże Hanoi

Koniec świata

- Przy trzech krążkach trzeba 7 ruchów (przy dwu — 3!)
- Gdy krążków jest 64 — trzeba $2^{64} - 1$ ruchów, czyli 18446744073709551615
- Zakładamy, że przeniesienie krążka zajmuje 1 sekundę.
- Rok ma $365 \times 24 \times 3600$ sekund (31536000)

Wieże Hanoi

Koniec świata

- Przy trzech krążkach trzeba 7 ruchów (przy dwu — 3!)
- Gdy krążków jest 64 — trzeba $2^{64} - 1$ ruchów, czyli 18446744073709551615
- Zakładamy, że przeniesienie krążka zajmuje 1 sekundę.
- Rok ma $365 \times 24 \times 3600$ sekund (31536000)
- Praca ta zajmie $2^{64}/31536000$ lat (prawie 585 miliardów lat)

Wieże Hanoi

Jak rozwiązać ten problem?

- Gdy krążek jest tylko jeden — problem nie istnieje.

Wieże Hanoi

Jak rozwiązać ten problem?

- Gdy krążek jest tylko jeden — problem nie istnieje.
- Dla dwu krążków problem jest banalny.

Wieże Hanoi

Jak rozwiązać ten problem?

- Gdy krążek jest tylko jeden — problem nie istnieje.
- Dla dwu krążków problem jest banalny.
- Dla trzech — można go podzielić na trzy zadania:

Wieże Hanoi

Jak rozwiązać ten problem?

- Gdy krążek jest tylko jeden — problem nie istnieje.
- Dla dwu krążków problem jest banalny.
- Dla trzech — można go podzielić na trzy zadania:
  1. Przeniesienie dwu „górnych” krążków na „trzeci patyczek”.

Wieże Hanoi

Jak rozwiązać ten problem?

- Gdy krążek jest tylko jeden — problem nie istnieje.
- Dla dwu krążków problem jest banalny.
- Dla trzech — można go podzielić na trzy zadania:
  1. Przeniesienie dwu „górnych” krążków na „trzeci patyczek”.
  2. Przeniesienie największego krążka na „patyczek drugi”.

Wieże Hanoi

Jak rozwiązać ten problem?

- Gdy krążek jest tylko jeden — problem nie istnieje.
- Dla dwu krążków problem jest banalny.
- Dla trzech — można go podzielić na trzy zadania:
  1. Przeniesienie dwu „górnych” krążków na „trzeci patyczek”.
  2. Przeniesienie największego krążka na „patyczek drugi”.
  3. Ponowne przeniesienie dwu krążków na największy...

Wieże Hanoi

Przypadek ogólny

1. Przenieśmy z A na C $N-1$ krążków.
2. Pozostały krążek (największy! — ale z czego to wynika?) przenieśmy z A na B.
3. Do pozostałych $(N-1)$ krążków zastosujmy powyższy algorytm (patyk B możemy wykorzystywać jako roboczy, bo na samym spodzie znajduje się krążek największy).
4. powyższą procedurę należy powtarzać aż do zakończenia zadania.

(Pierwszy patyczek nazwaliśmy A, drugi — B a trzeci — C.)

Wieże Hanoi

Procedura (rekurencyjna)

Procedura **przenieś** N (krążków) **z** X **na** Y **używając** Z:

1. Jeśli $N = 1$ to wypisz „$X \to Y$”;
2. w przeciwnym razie (jeżeli $N > 1$) wykonaj co następuje:
   - 2.1 wywołaj **przenieś** $N - 1$ **z** X **na** Z **używając** Y;
   - 2.2 wypisz „$X \to Y$”;
   - 2.3 wywołaj **przenieś** $N - 1$ **z** Z **na** Y **używając** X;
3. wróć;

Zapis symboliczny $A \to B$ oznacza „weź krążek z *patyka* oznaczonego A i przenieś go na *patyk* oznaczony B”
Procedura służy jedynie do wypisywania instrukcji dla „operataora—człowieka”.

Wieże Hanoi

Realizacja

Start

przenieść 3 **z** A **na** B **używając** C

1. **przenieść** 2 **z** A **na** C **używając** B
   - 1.1 **przenieść** 1 **z** A **na** B **używając** C
     - 1.1.1 $A \to B$
   - 1.2 $A \to C$
   - 1.3 **przenieść** 1 **z** B **na** C **używając** A
     - 1.3.1 $B \to C$
2. $A \to B$
3. **przenieść** 2 **z** C **na** B **używając** A
   - 3.1 **przenieść** 1 **z** C **na** A **używając** B
     - 3.1.1 $C \to A$
   - 3.2 $C \to B$
   - 3.3 **przenieść** 1 **z** A **na** B **używając** C
     - 3.3.1 $A \to B$

Podział...

Przykład

procedura **znajdź-min-i-max-w** L;

1. Jeżeli lista składa się z jednego elementu to nadaj MAX i MIN właśnie jego wartość, jeśli lista składa się z dwu elementów to nadaj MIN wartość mniejszego z nich, a MAX większego;
2. W przeciwnym razie wykonaj co następuje:
   - 2.1 Podziel listę na połowy L1 i L2;
   - 2.2 wykonaj **znajdź-min-i-max-w** L1 umieszczając otrzymane wartości w MIN1 i MAX1
   - 2.3 wykonaj **znajdź-min-i-max-w** L2 umieszczając otrzymane wartości w MIN2 i MAX2
   - 2.4 nadaj MIN mniejszą wartość z MIN1 i MIN2
   - 2.5 nadaj MAX większą wartość z MAX1 i MAX2
3. zakończ z wartościami MIN i MAX

Zachłanne algorytmy

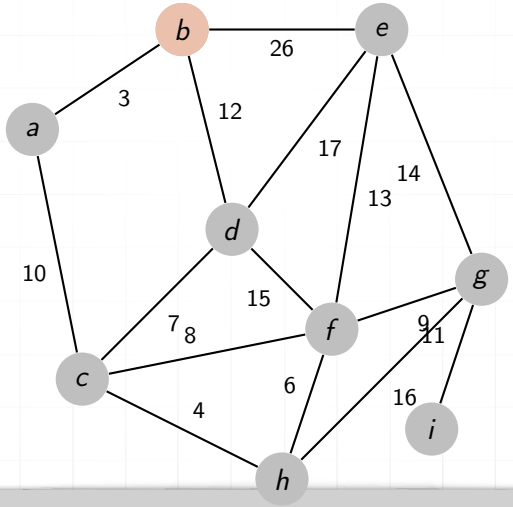
- Czasami zadanie sprowadza się do znalezienia rozwiązania najlepszego w jakimś sensie

Popatrzmy na przykład: Mamy sieć miast i leniwego przedsiębiorcę, któremu zlecono ułożenie torów zapewniających połączenie możliwość przejazdu z dowolnego miasta do każdego innego. Innych warunków nie postawiono...

Koszt położenia torów jest proporcjonalny do odległości pomiędzy miastami. Leniwy przesiębiorca chce rozwiązać problem jak najtańszym kosztem.

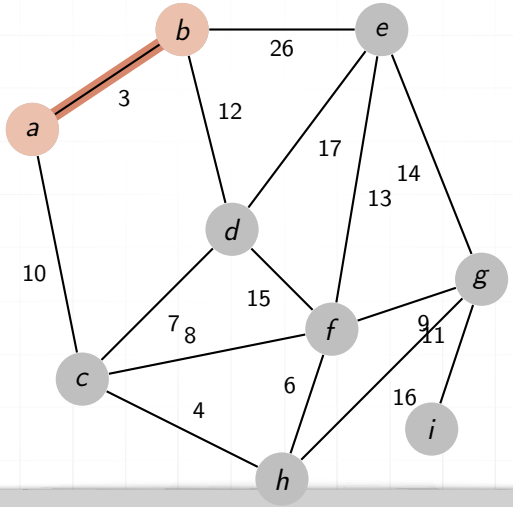
Zachłanne algorytmy

Przykład



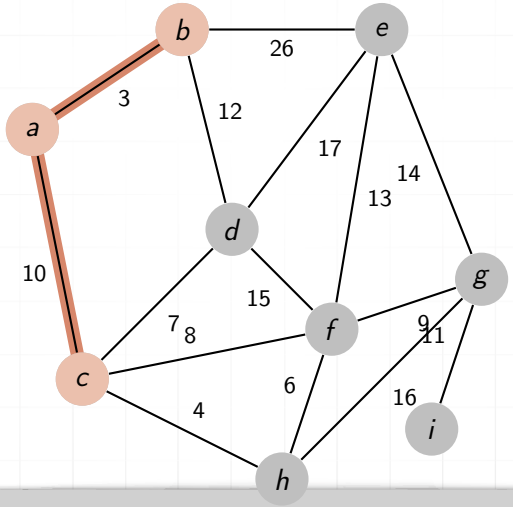
Zachłanne algorytmy

Przykład



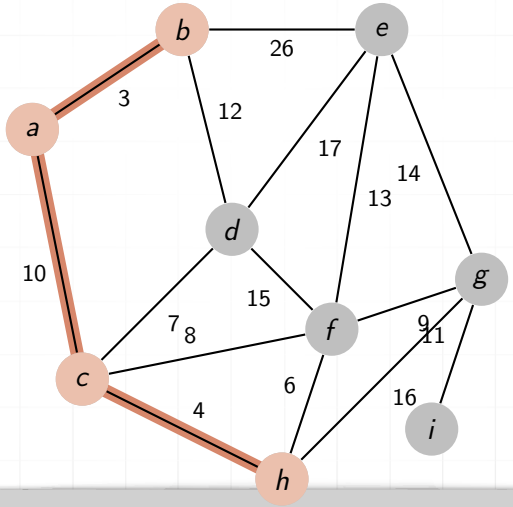
Zachłanne algorytmy

Przykład



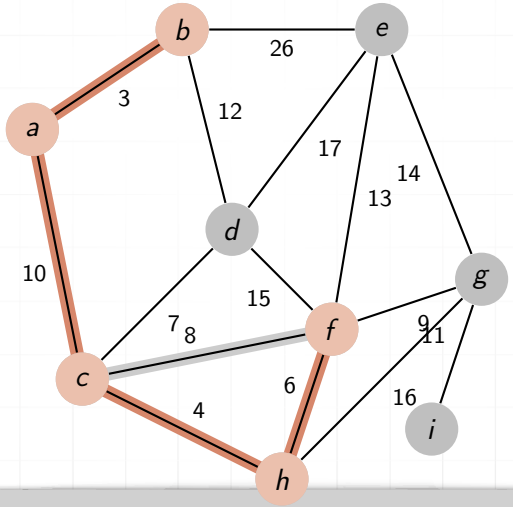
Zachłanne algorytmy

Przykład



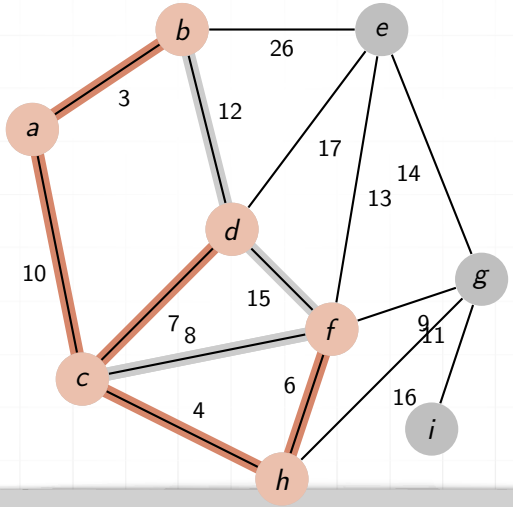
Zachłanne algorytmy

Przykład



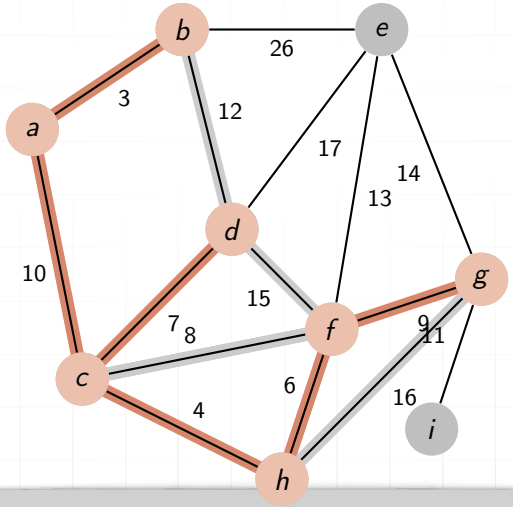
Zachłanne algorytmy

Przykład



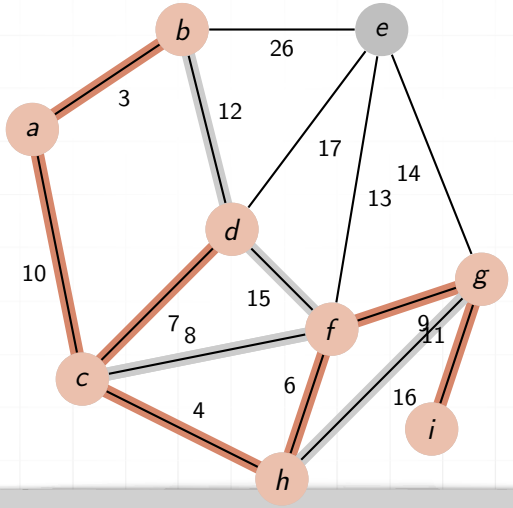
Zachłanne algorytmy

Przykład



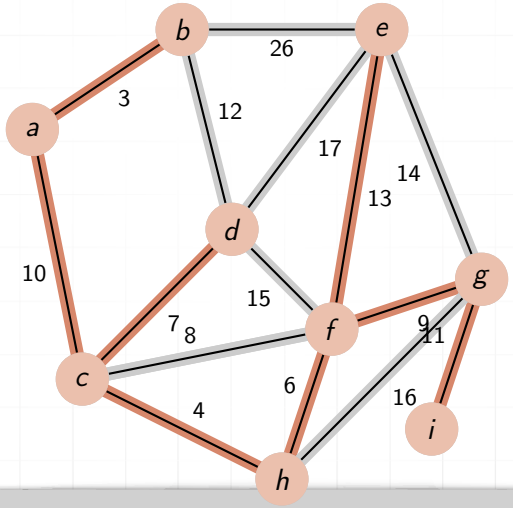
Zachłanne algorytmy

Przykład



Zachłanne algorytmy

Przykład



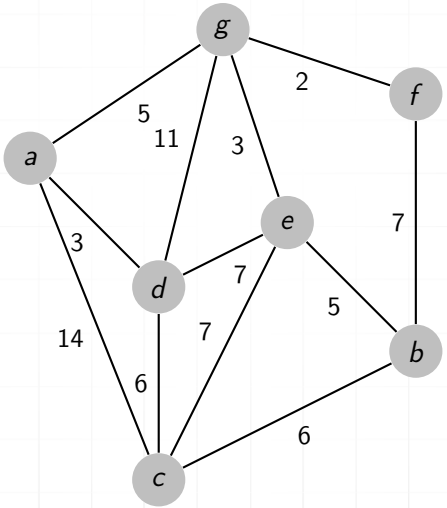
Zachłanne algorytmy

Czemu taka nazwa?

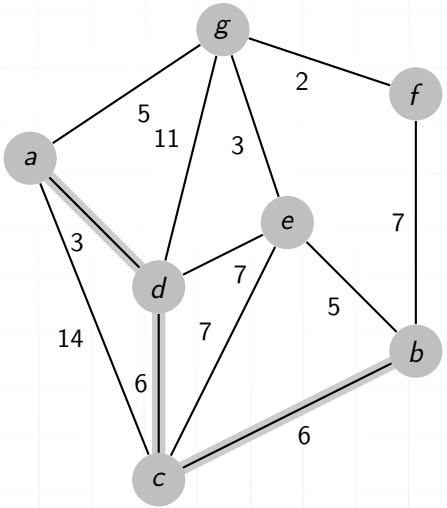
Planowanie dynamiczne

- Załóżmy, że mamy (podobnie jak w zadaniu poprzednim) szereg miast.
- W poprzednim zadaniu mieliśmy znaleźć taki układ połączeń, który (a) jest najtańszy i (b) pozwala na przejazd z dowolnego miasta do każdego innego.
- Teraz mamy znaleźć najtańszą (najkrótszą) drogę z miasta A do miasta B korzystając z istniejącej sieci połączeń.

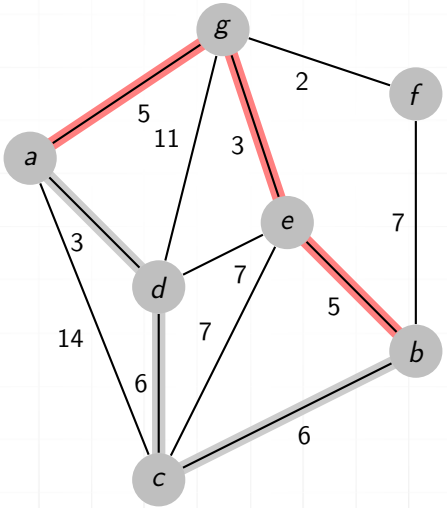
Planowanie dynamiczne



Planowanie dynamiczne



Planowanie dynamiczne



Planowanie dynamiczne

- Jak widać metoda „zachłanna” nie daje najlepszego rozwiązania.
- Zatem potrzebna będzie nieco inna metoda postępowania — znacznie bardziej „przewidująca”.
- Aby dojść z A do B w pierwszym kroku mam trzy możliwości:
  1. przejść do C
  2. przejść do D
  3. przejść do E
- koszt pierwszej to 5 + koszt przejścia z C do B
- koszt drugiej to 3 + koszt przejścia z D do B
- koszt trzeciej to 14 + koszt przejścia z G do B

Programowanie dynamiczne

Na czym powinien polegać optymalny wybór?

- Oznaczmy przez L(x) koszt przejścia z węzła x do węzła B
- na każdym etapie podróży, gdy możemy przejść z węzła V do węzłów C_1, C_2,... C_N wybieramy taką drogę aby

$$\text{odległość-z-}V\text{-do-}C_K + L(C_K)$$

była minimalna

Można to zapisać tak:

$$L(V) = min_K \text{odległość-z-}V\text{-do-}C_K + L(C_K)$$

Programowanie dynamiczne

Taki zapis sugeruje również tu zastosowanie rekursji — ale ostrzegam! jest to niebezpieczne...