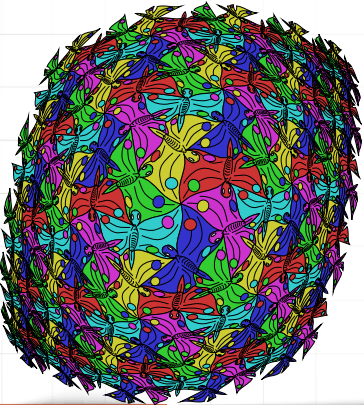


Metody algortmiczne (Algorytmy Część IV)

wer. 12 z drobnymi modyfikacjami!

Wojciech Myszka

2023-12-04 12:31:07 +0100

Politechnika Wrocławska

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

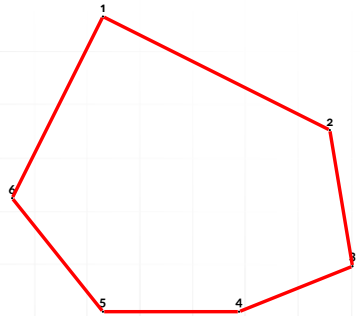
Czyli pętla

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**

Przegląd

Przykład

- Dany jest prosty wielokąt wypukły.
- Należy znaleźć takie dwa punkty na jego obwodzie, które dzieli największa odległość.

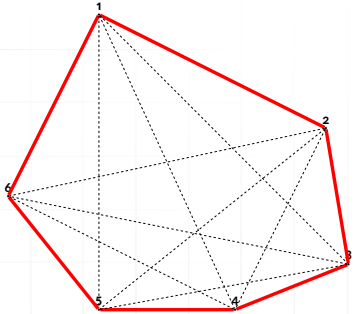


Jak to rozwiązać?

Przegląd

Przykład

- Dany jest prosty wielokąt wypukły.
- Należy znaleźć takie dwa punkty na jego obwodzie, które dzieli największa odległość.



Jak to rozwiązać?

Przykład cd

	1	2	3	4	5	6
1	0	11,2	15,6			
2	11,2	0				
3	15,6		0			
4				0	6	
5				6	0	6,4
6					6,4	0

Wierzchołki:
1 (4,13)
2 (14,8)
3 (15,2)
4 (10,0)
5 (4,0)
6 (0,5)

Przykład cd

Najprostszy przypadek

1. max = 0
2. **for** i = 1 to 6 **do**
   - 2.1 **for** j = 1 to 6 **do**
   - 2.2 **if** d[i, j] > max **then**
   - 2.3 max = d[i,j]

Przykład cd

Najprostszy przypadek

1. max = 0
2. **for** i = 1 to N **do**
   - 2.1 **for** j = 1 to N **do**
   - 2.2 **if** d[i, j] > max **then**
   - 2.3 max = d[i,j]

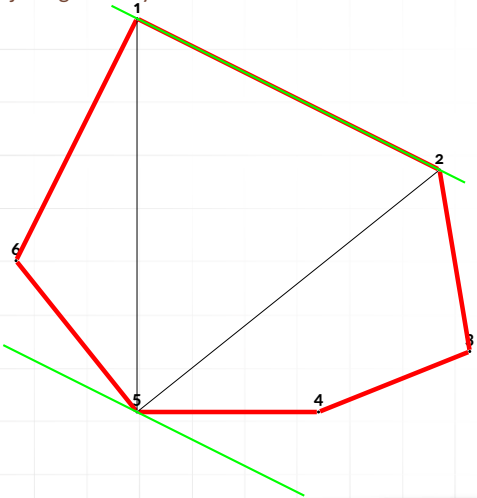
Przykład cd

Najprostszy przypadek

1. max = 0
2. **for** i = 1 to N **do**
   - 2.1 **for** j = 1 to i — 1 **do**
   - 2.2 **if** d[i, j] > max **then**
   - 2.3 max = d[i,j]

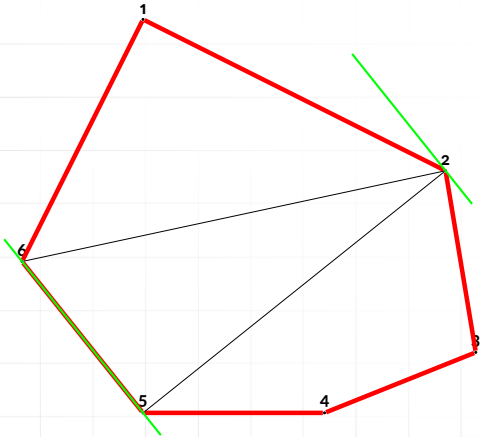
Przykład cd

„Podejście geometryczne”



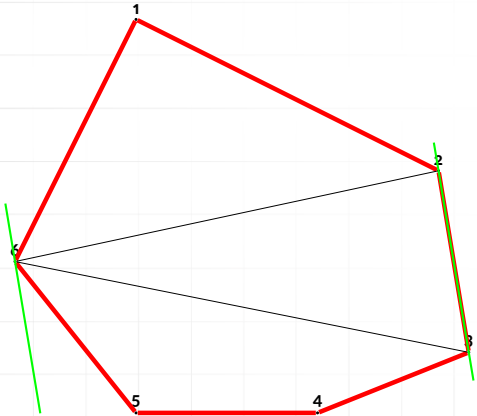
Przykład cd

„Podejście geometryczne”



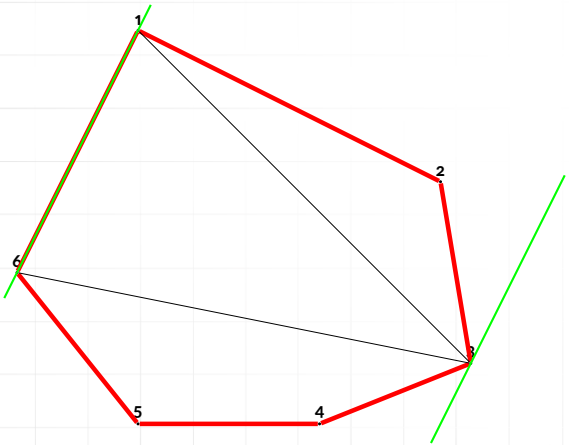
Przykład cd

„Podejście geometryczne”



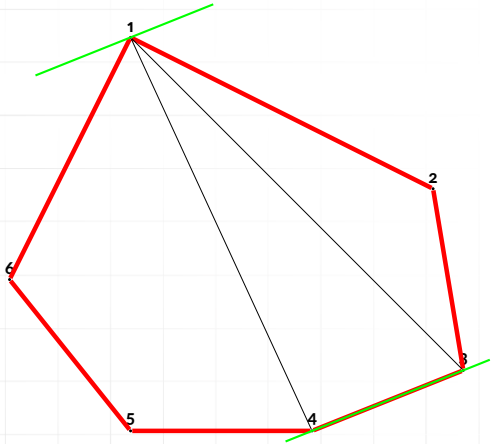
Przykład cd

„Podejście geometryczne”



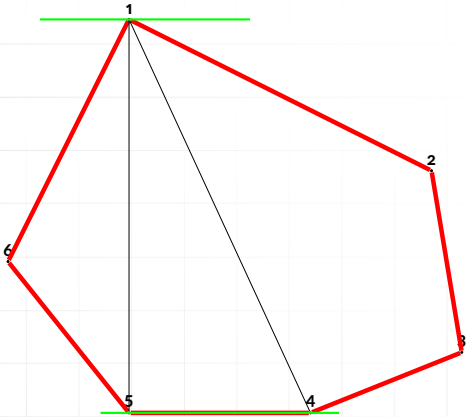
Przykład cd

„Podejście geometryczne”



Przykład cd

„Podejście geometryczne”



Dziel i zwyciężaj

Podział na mniejsze zadania

Idea bardzo prosta

- Gdy nie możemy rozwiązać „dużego” zadania — czasami można je podzielić na kilka mniejszych
- Jeżeli te są za duże — dzielimy je dalej…

Podział…

Przykład

Mamy znaleźć maksimum i minimum z (bardzo długiej?) listy liczb oznaczonej jako *L*

Jedną metodę już znamy: jest dosyć prosta…

Czy można inaczej?

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 L_1 i L_2;
   - 2.2 wykonaj **znajdź-min-i-max-w** L_1 umieszczając otrzymane wartości w MIN_1 i MAX_1
   - 2.3 wykonaj **znajdź-min-i-max-w** L_2 umieszczając otrzymane wartości w MIN_2 i MAX_2
   - 2.4 nadaj MIN mniejszą wartość z MIN_1 i MIN_2
   - 2.5 nadaj MAX większą wartość z MAX_1 i MAX_2
3. zakończ z wartościami MIN i MAX

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

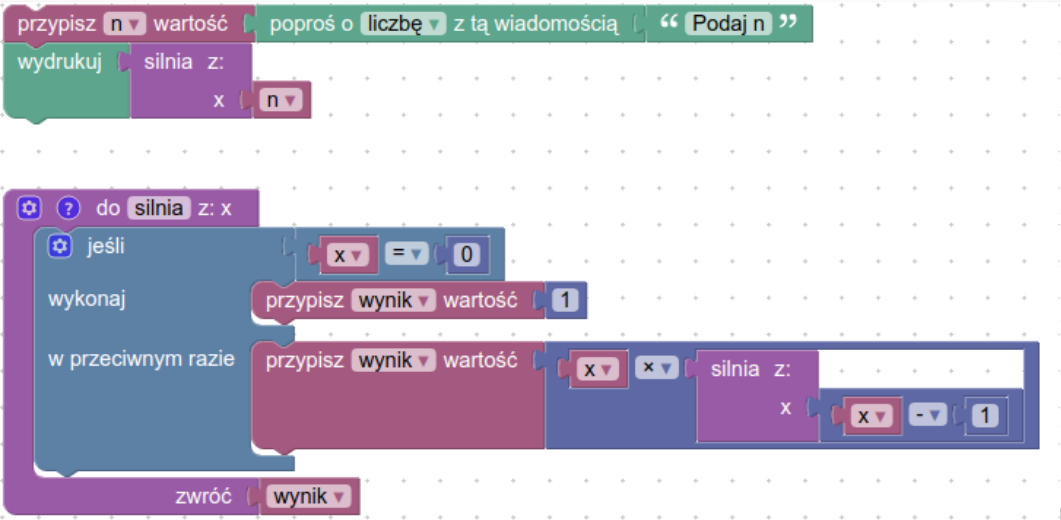
$n! = 1 \times 2 \times 3 \times \cdots \times n$

Wersja inna

$0! = 1$

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

Silnia w blockly



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
   $silnia \leftarrow silnia \times i$
4. Koniec algorytmu.

Silnia

„Schematy blokowe”

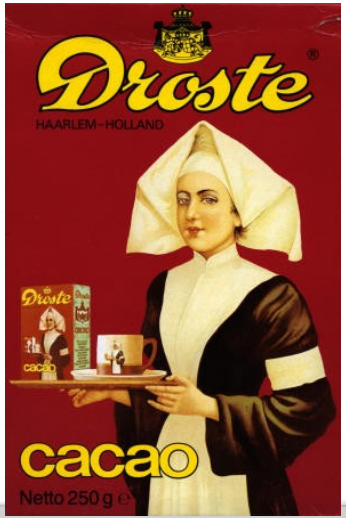
Wersja klasyczna

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

Wersja rekurencyjna

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

Rekurencja na obrazkach: Droste Effect



Rekurencja — Escher

"Prententententoonstelling" (M.C. Escher 1956)

Rekurencja — Escher

"Prentententoonstelling" (M.C. Escher 1956)

Rekurencja — Escher

"Prentententoonstelling" (M.C. Escher 1956)

Rekurencja — Escher

"Prententententoonstelling" (M.C. Escher 1956)

Rekurencja — Escher

“Prentententoonstelling” (M.C. Escher 1956)

Rekurencja — Escher

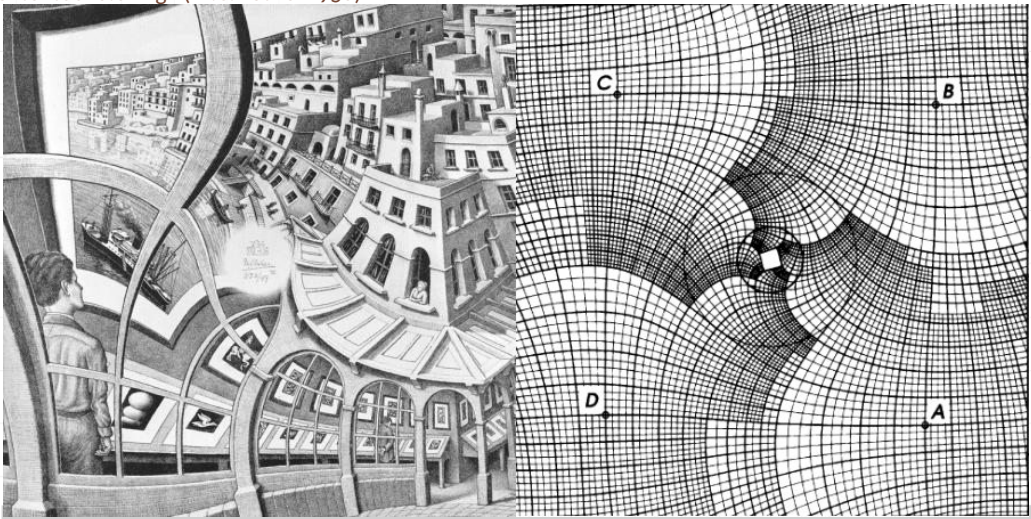
"Prentententoonstelling" (M.C. Escher 1956)

Rekurencja — Escher

"Prentententoonstelling" (M.C. Escher 1956)

Rekurencja — Escher

"Prentententoonstelling" (M.C. Escher 1956)



Rekurencja — Escher

"Prententententoonstelling" (M.C. Escher 1956)

Na podstawie http://escherdroste.math.leidenuniv.nl/

Ciąg Fibonacciego

Schemat

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

Ciąg Fibonacciego

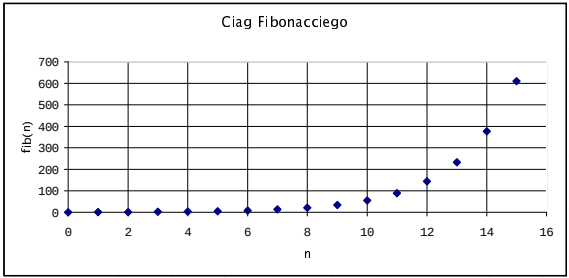
Schemat

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	



Ciąg Fibonacciego

Schemat

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

Nierekurencyjnie

$$F(k) = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^k - \left(\frac{1-\sqrt{5}}{2} \right)^k \right]$$

Więcej na temat liczby Fibonacciego tu: [1].

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(o,n)=n

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

Zadanie domowe

- Porównać ze schematem blokowym („metoda Euklidesa”), który był prezentowany wcześniej
- Zaprogramować w blockly

Symbol Newtona

Wersja „normalna”

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

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

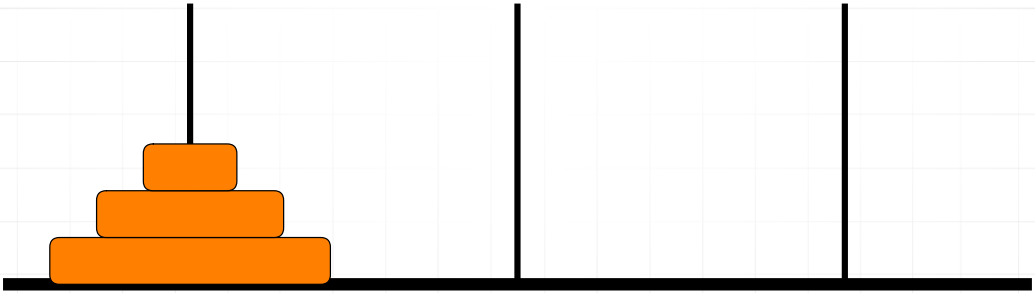
Wersja „rekurencyjna”

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

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

Wieże Hanoi

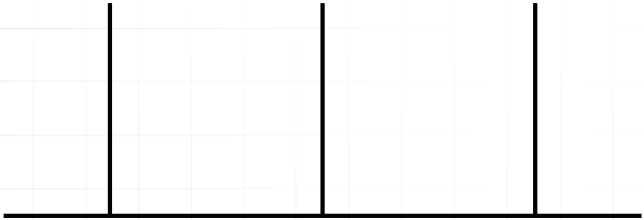
Prosta zabawka dziecięca — na patyku nanizanych jest pewna liczba krążków tak, że na większym zawsze leży krążek mniejszy. Zadaniem naszym jest umieszczenie wszystkich krążków na sąsiednim „patyku” (korzystając z jednego tylko „patyka” pomocniczego) w tej samej kolejności. Podczas każdego ruchu pamiętać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



Wieże Hanoi

Przykład

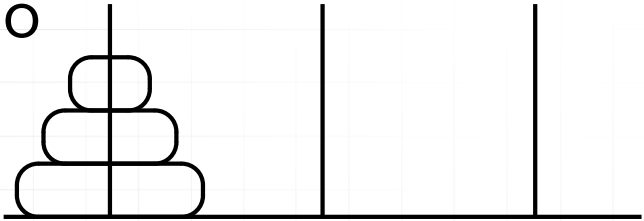
Trzy krążki



Wieże Hanoi

Przykład

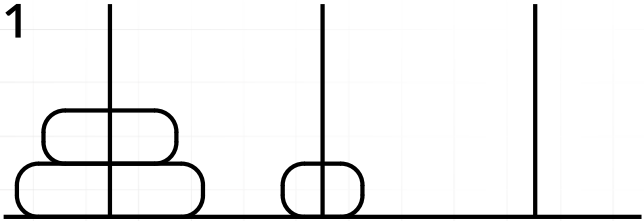
Trzy krążki



Wieże Hanoi

Przykład

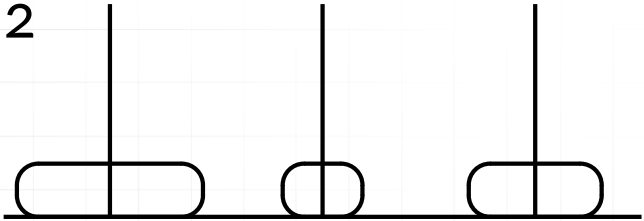
Trzy krążki



Wieże Hanoi

Przykład

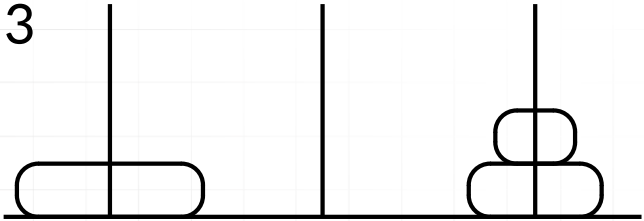
Trzy krążki



Wieże Hanoi

Przykład

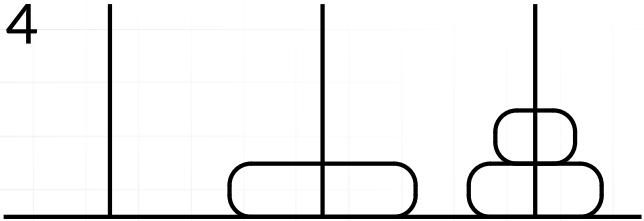
Trzy krążki



Wieże Hanoi

Przykład

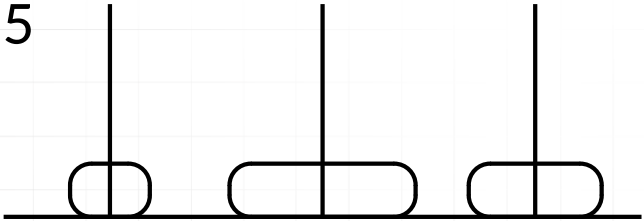
Trzy krążki



Wieże Hanoi

Przykład

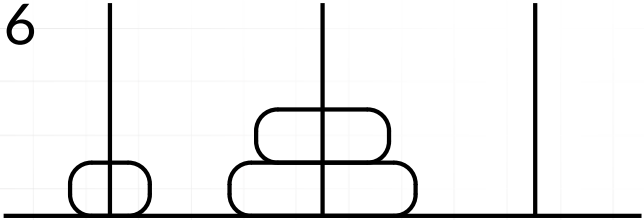
Trzy krążki



Wieże Hanoi

Przykład

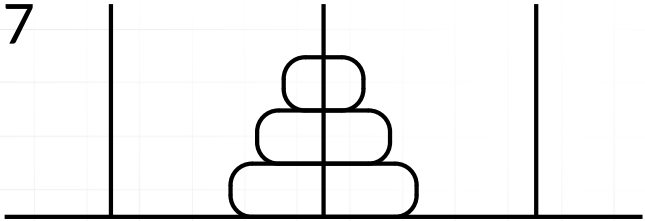
Trzy krążki



Wieże Hanoi

Przykład

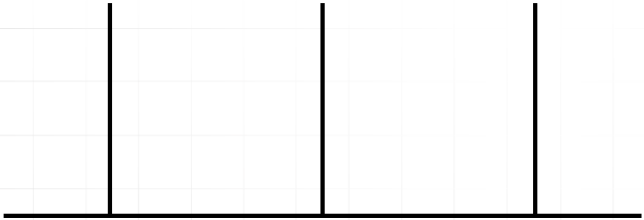
Trzy krążki



Wieże Hanoi

Przykład

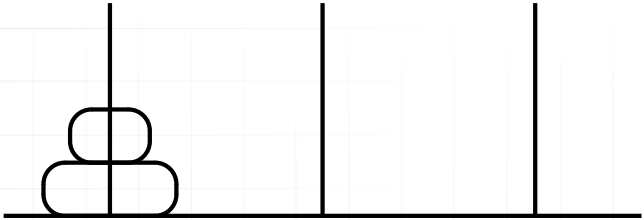
Dwa krążki



Wieże Hanoi

Przykład

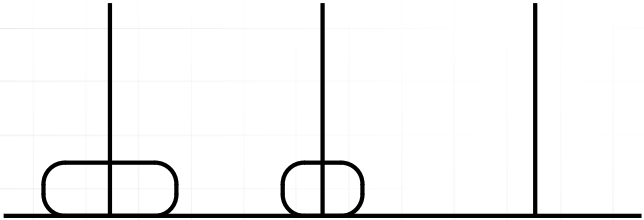
Dwa krążki



Wieże Hanoi

Przykład

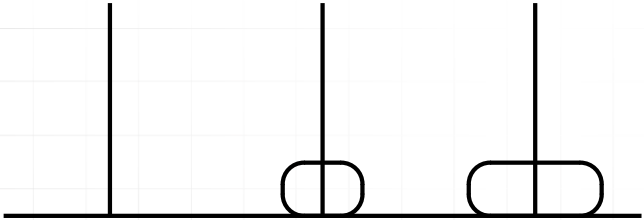
Dwa krążki



Wieże Hanoi

Przykład

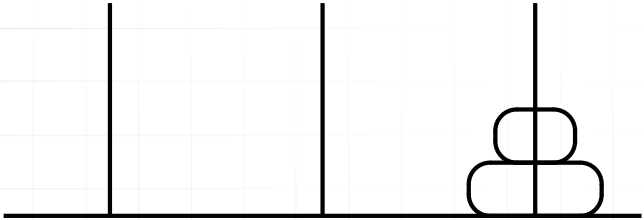
Dwa krążki



Wieże Hanoi

Przykład

Dwa krążki



Wieże Hanoi I

Czemu taka nazwa?

„W wielkiej świątyni Benares, 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.

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”.

Wieże Hanoi II

Czemu taka nazwa?

(Jest też wersja mówiąca o świątyni w Hanoi — stąd nazwa.)

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 „operatora–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 $A \to B$

Zachłanne algorytmy

i przetargi kolejowe

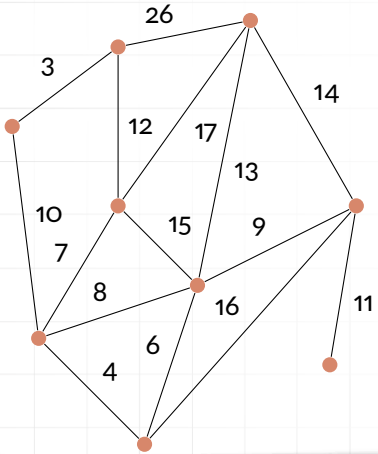
- Czasami zadanie sprowadza się do znalezienia rozwiązania najlepszego w jakimś sensie

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

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

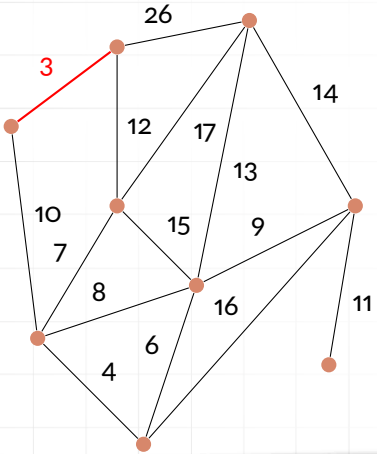
Zachłanne algorytmy

i przetargi kolejowe



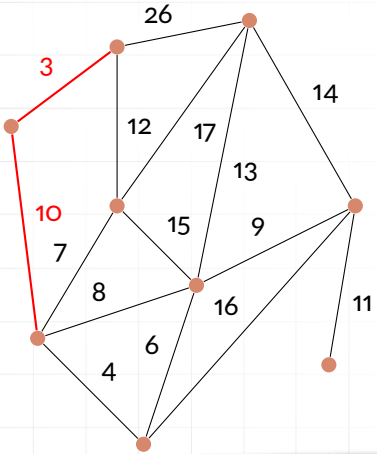
Zachłanne algorytmy

i przetargi kolejowe



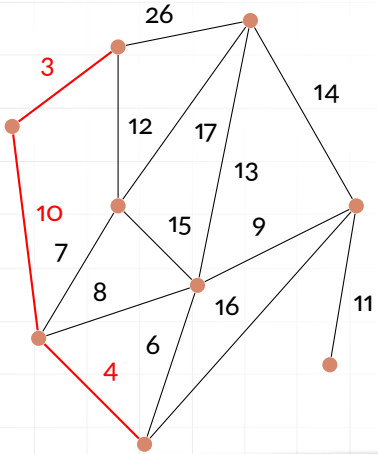
Zachłanne algorytmy

i przetargi kolejowe



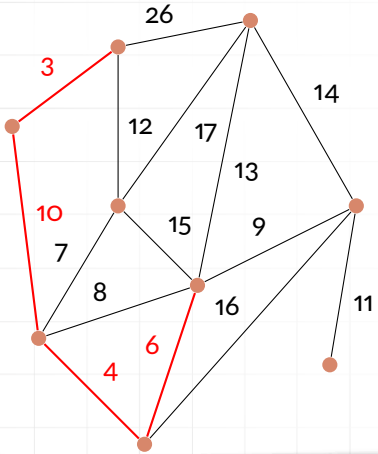
Zachłanne algorytmy

i przetargi kolejowe



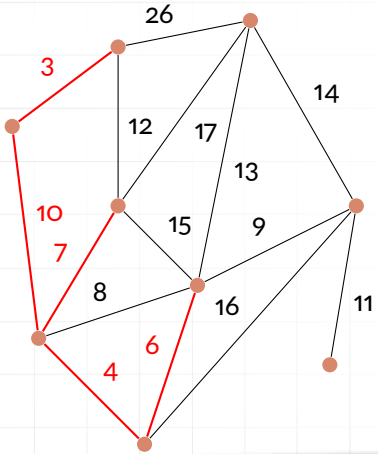
Zachłanne algorytmy

i przetargi kolejowe



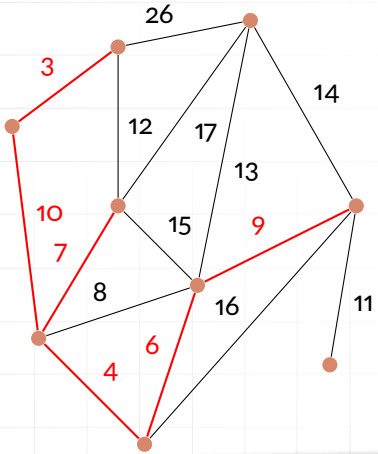
Zachłanne algorytmy

i przetargi kolejowe



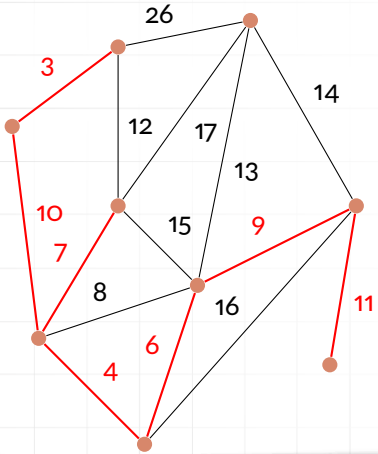
Zachłanne algorytmy

i przetargi kolejowe



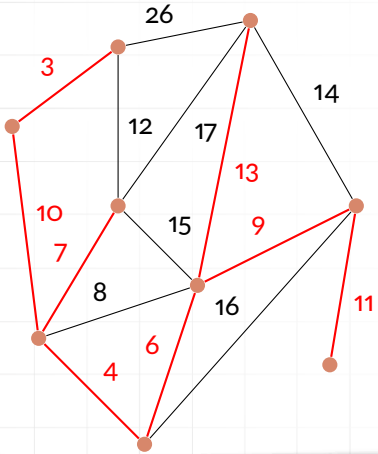
Zachłanne algorytmy

i przetargi kolejowe



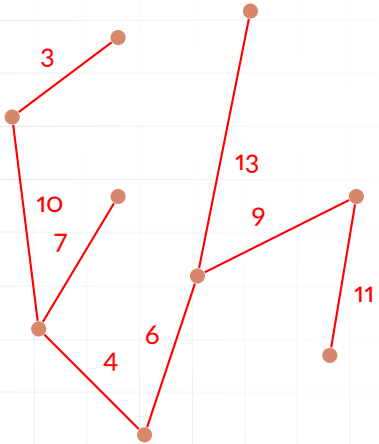
Zachłanne algorytmy

i przetargi kolejowe



Zachłanne algorytmy

i przetargi kolejowe



Taki graf nazywa się *minimalnym drzewem rozpinającym*

Zachłanne algorytmy

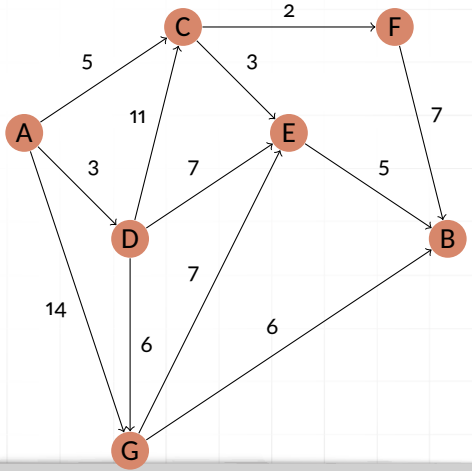
Czemu taka nazwa?

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

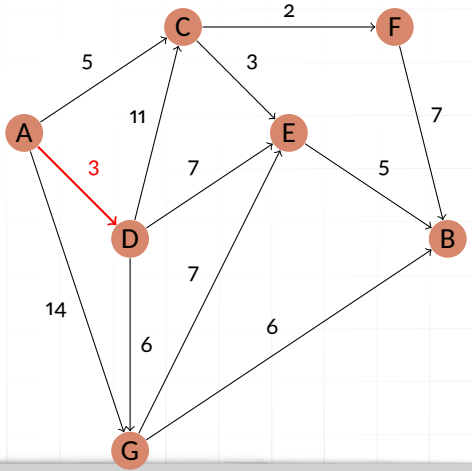
Przykład



Planowanie dynamiczne

Przykład

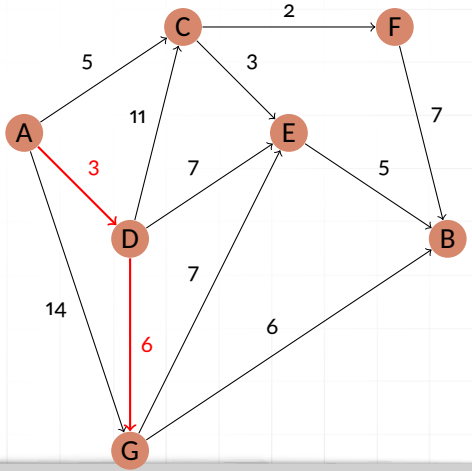
Rozwiązanie zachłanne



Planowanie dynamiczne

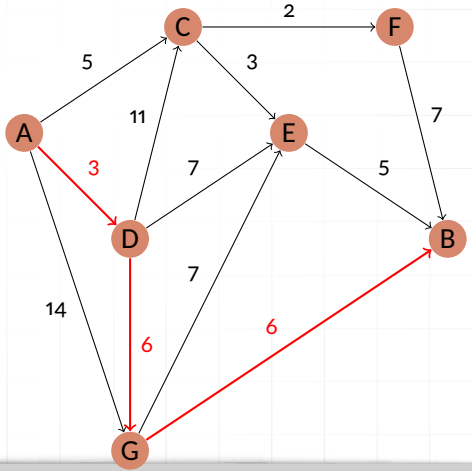
Przykład

Rozwiązanie zachłanne



Planowanie dynamiczne

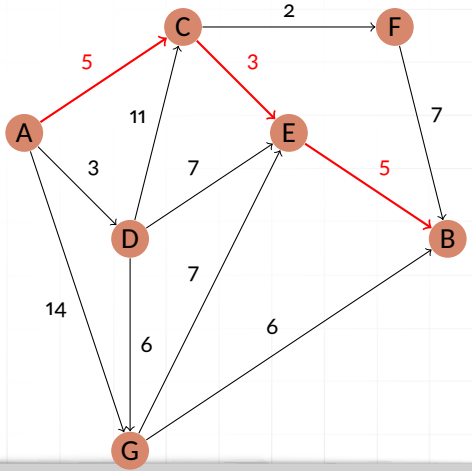
Przykład



Rozwiązanie zachłanne
Koszt całkowity 15

Planowanie dynamiczne

Przykład



Rozwiązanie optymalne
Koszt całkowity 13

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 G
- 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 (końcowego)
- na każdym etapie podróży, gdy możemy przejść z węzła V do węzłów $C_1, C_2, \ldots C_N$ 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…

Do obejrzenia

Mirosław Zelent.
Tajemniczy ciąg fibonacciego. złota liczba. boska proporcja, December 2013.
http://www.youtube.com/watch?v=wb7kPaM8cfg&feature=youtube_gdata_player.