Algorytmy zachłanne
- często nie działa
- trzeba dowieść poprawności
- czasem aproksymacje rozwiązanie

Przykład
Wydawanie monet

1, 2, 5, 10, 20, 50, ⟨...⟩

R = 4,98
↓ 2
R = 2,98 ~sprowadzamy problem do mniejszego
↓ 2
R = 0,98
↓ 0,50
0,48
↓ 0,20
0,28
↓ 0,20
0,08
↓ 0,05
↓ 0,02
↓ 0,01
R=0

8 monet

Dlaczego nie można lepiej?

Lemat: Istnieje rozw. optymalne z największą dostępną monetą

Dla 1, 2, 4, ... 2^k:

Lemat: Rozwiązanie, które zawiera 2 monety tego samego nominału, nie jest optymalne

D-d
Weźmy rozw. optymalne, w którym tej monety nie wzięto.

Ogólny schemat: wybór:
1) przekształcamy optymalne na nasze
2) pokazujemy, że nie jest optymalne

→ 2)
$R \geq 2^k$

$\underbrace{1\,1\,1\,1\,1\,1}_{2^k}\,0\,1\,1$

$\sum_{i=0}^{k-1} c_i 2^i \quad c_i \leq 1$

$\sum_{i=0}^{k-1} c_i 2^i < 2^k$ ↯

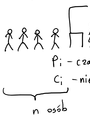
→ nie dla każdego ciągu nominałów działa

1, 4, 5 R=12 5, 5, 1, 1 4, 4, 4
1, 4, 9 R=12 9, 1, 1, 1 4, 4, 4

1 2 5 10 20

Przykład 2.

Problem pani z dziekanatu



p_i - czas obsługi
c_i - niecierpliwość
n osób

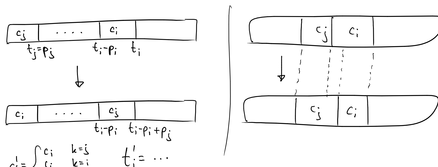
ułożenie: $t_i \cdot c_i$, t_i - czas obsłużenia
minimalizacja $\sum c_i t_i$ $\quad t_i = \sum_{j=0}^{k_i} p_{k_i}$
k_i - kolejność

$w_i =$ $p_i \cdot c_i$ $\quad p_i + c_i$ $\quad \frac{p_i}{c_i}$

$p_i \cdot \sum_{j \neq i} c_j$
$k_j \geq k_i - 1$

Lemat: Niech i ~ osoba o najmniejszym $\frac{p_i}{c_i}$. Wtedy istnieje rozwiązanie optymalne, w którym ta osoba jest pierwsza

Weźmy dowolne rozw. optymalne, osoba i nie jest pierwsza



$c'_k = \begin{cases} c_i & k = j \\ c_j & k = i \\ c_k & \text{wpp} \end{cases}$ $\quad t'_i = \dots$

(rozw. opt.:)
$0 \leq \sum c_i t_i - \sum c'_i t_i$

Przykład 3: problem szeregowania

N - l. zadań
d_i - deadline (każde zadanie trwa 1)
z_i - zysk za zadanie

(posortuj zadania po deadline)
for $t \in [N..1]$ (szybko)
Q.add(filter (deadline = t) zadania)
yield <$> (Q.pop)

Lemat: Istnieje rozwiązanie optymalne, w którym występuje element z największym deadlinem o największym zysku (spośród tych o tym samym deadlinie) wykonany jako ostatni

Weźmy rozw. optymalne
1) pewien element występuje
1.1) jako ostatni □
1.2) gdzie indziej → zamieniamy go z ostatnim □
2) nie występuje
2.1) w czasie N nie ma zadania
Dokładamy zadanie, mamy lepsze rozwiązanie ↯
2.2) jest zadanie
Ma niewiększy zysk, zamieniamy, mamy
niegorsze rozwiązanie □

MST: Algorytmy:

Kruskala

Dijkstry ? Prima

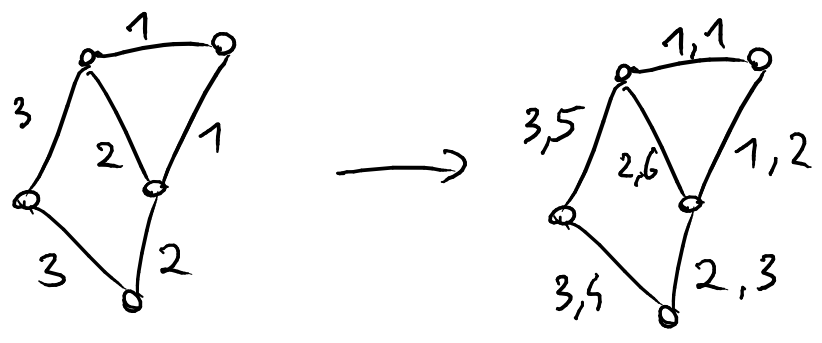
Boruvki

Boruvki

↳ krawędzie mają mieć różne wagi

↳ dodajemy drugą współrzędną, porządek leksykograficzny

MST jest jednoznacznie wyznaczone



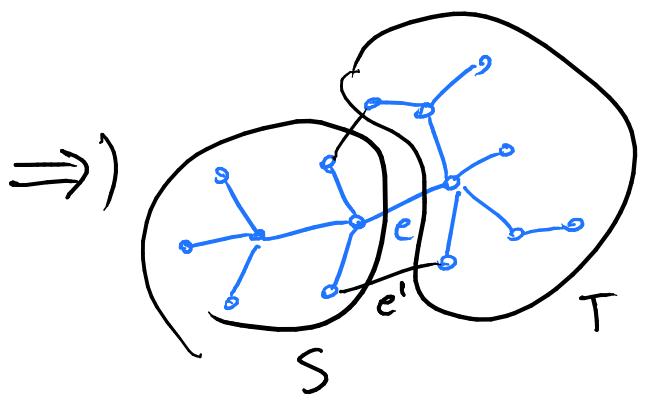
Boruvka → zrównoleglanie !

Boruvka

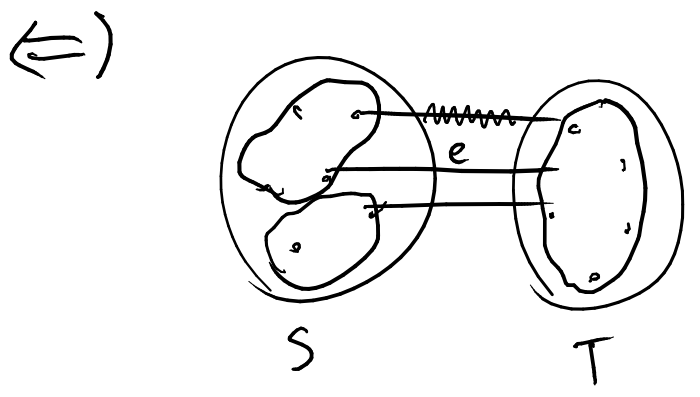
- każdy wierzchołek wybiera najmniejszą sąsiadującą krawędź
- powstałe spójne składowe zamieniamy w wierzchołki
- powtarzamy na nowym grafie jeśli $|V'| > 1$

Cut property

$e \in MST \Leftrightarrow \exists S,T \; S \cup T = V$. e - najtańsza krawędź między S i T



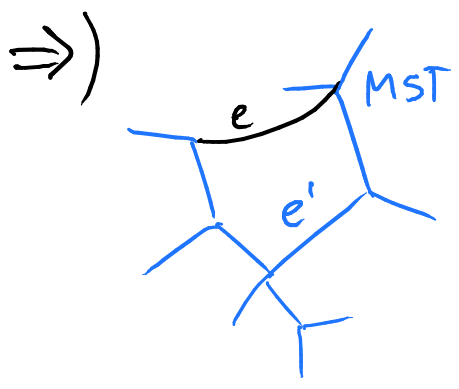
e - najtańsza, inaczej zastępujemy e e' i mamy lepsze MST



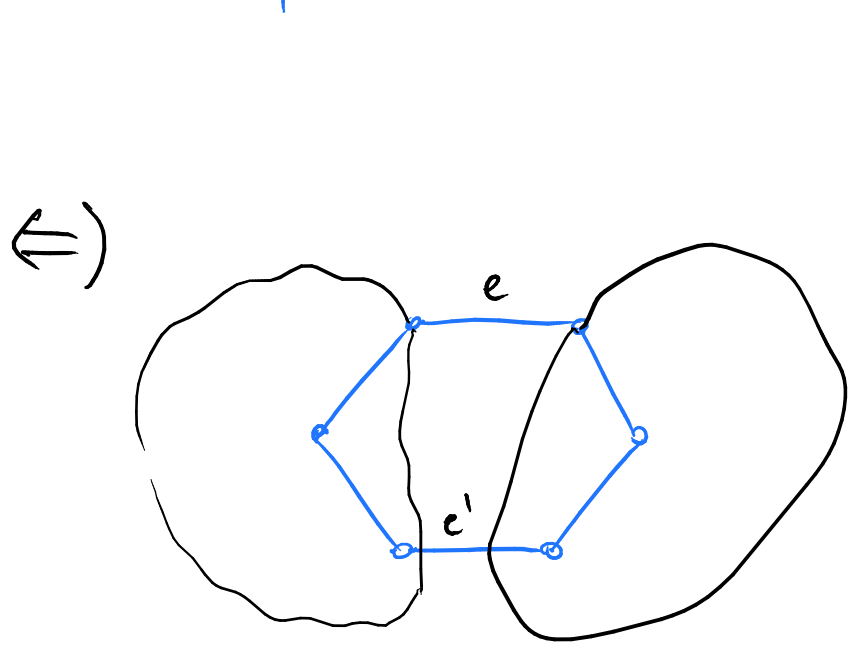
Cycle?

Circle property

$e \notin MST \Leftrightarrow \exists$ cykl. e - najcięższa na cyklu

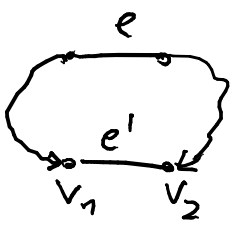


$e' < e \Rightarrow$ mamy lepsze MST ↯



Załóżmy $e \in MST$

Wtedy na cyklu $\exists v_1 \xrightarrow{e'} v_2$. ($\rightarrow$ MST)



$e \leftarrow e'$, lepsze MST ↯

(Dowieść poprawności Kruskala, Prima, Boruvki)