ASD - Wykład 7

Problem komiwojażera

Dane: $C = \{0, \dots, n-1\}$ - zbiór miast

$d : C \times C \to \mathbb{R}$ - metryka nad C

Zadanie: Znaleźć trasę zaczynającą się w mieście 0, prowadzącą przez wszystkie inne miasta (przez każde dokładnie raz) i wracającą do miasta 0 o minimalnej sumarycznej długości

Algorytm brute-force

Spróbuj każdej kolejności odwiedzania miast

$O(n \cdot n!)$ złożoność czasowa

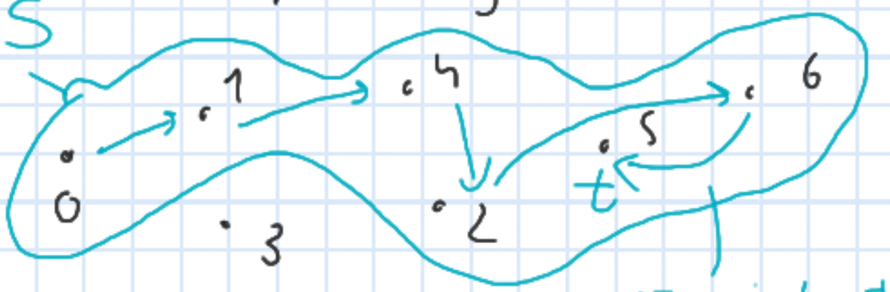
$O(n)$ złożoność pamięciowa

Algorytm dynamiczny

S - podzbiór miast, taki że $0 \in S$

$t \in S$ - miasto

$f(S,t)$ = długość (w sensie d) najkrótszej trasy z miasta 0 do miasta t prowadzącej przez wszystkie miasta z S



długość tej trasy to $f(S,t)$

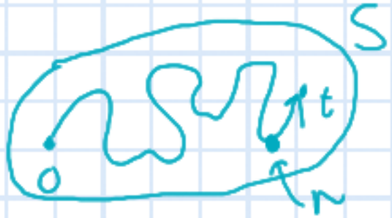
Rozwiązanie

$$\min_{t \in \{1,\dots,n-1\}} \left(f(C,t) + d(t,0) \right)$$

Sformułowanie rekurencyjne dla f

$f(\{0\}, 0) = 0$

$$f(S, t) = \min_{r \in S - \{t\}} \left(f(S - \{t\}, r) + d(r, t) \right)$$



Złożoność

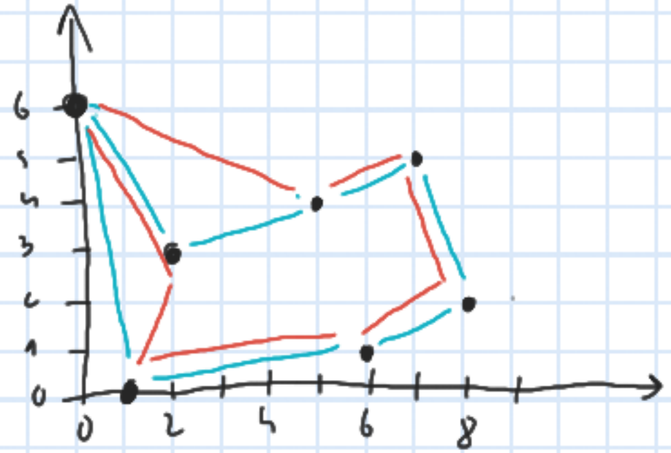
$\Theta(n^2 \cdot 2^n)$ — czasowa

$\Theta(n \cdot 2^n)$ — pamięciowa

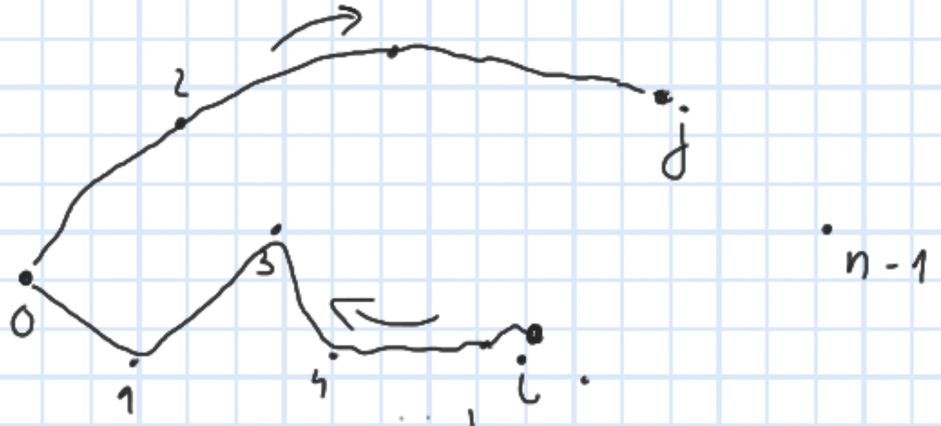
Bitoniczny problem komiwojażera

Wersja problemu w której

- miasta to punkty w $\mathbb{R}^2$ (żadne dwa miasta nie mają tej samej wsp. x)
- miasto 0 ma najmniejszą wsp. x
- szukamy trasy, która przebiega z lewa na prawo i z powrotem (kierunek na osi x zmieniamy raz)



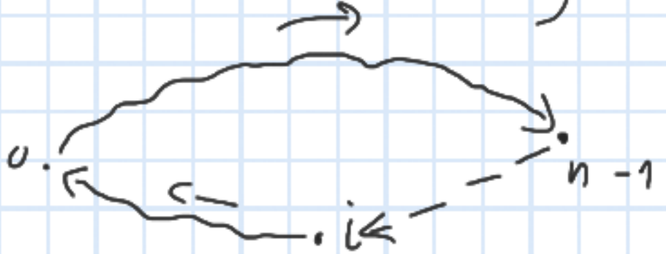
Algorytm dynamiczny



$f(i,j)$ $i<j$ = (minimalny) koszt ścieżek z 0 do i oraz z 0 do j, które używają łącznie wszystkich miast $\{0,\dots,j\}$, ale żadnego nie powtarzają

Rozwiązanie

$$\min_i \left(f(i, n-1) + d(i, n-1) \right)$$

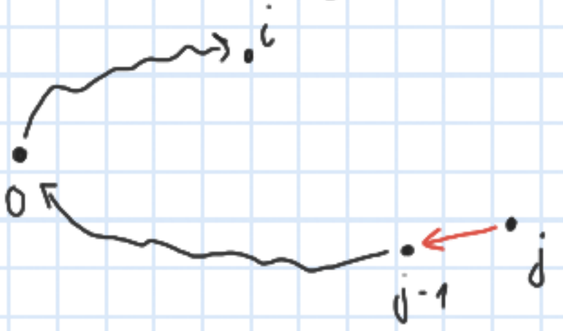


Zapis rekurencyjny funkcji f

$$f(0,1) = d(0,1)$$

Następnie rozważamy dwa przypadki:

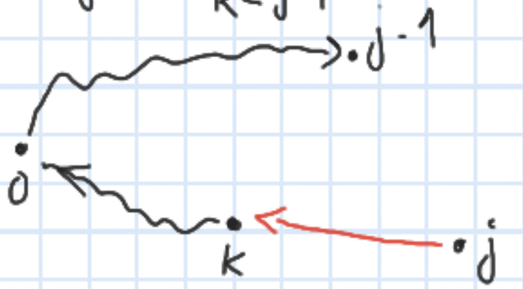
a)



$$f(i,j) = f(i,j-1) + d(j-1,j)$$

$i < j-1$

b) $$f(j-1,j) = \min_{k<j-1} f(k,j-1) + d(k,j)$$



Implementação

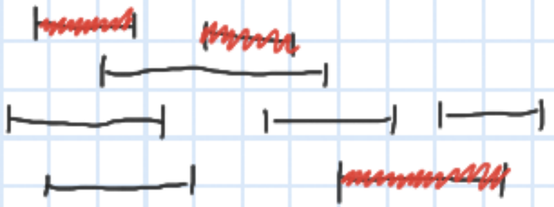
```
D[i][j] = d(i, j)
F       = [ [∞ for j in range(n)] for i in range(n) ]

def tspf(i, j, F, D):
    if F[i][j] ≠ ∞ : return F[i][j]
    if i = j-1 :
        best = ∞
        for k in range(j-1):
            best = min(best, tspf(k, j-1, F, D) + D[k][j])
        F[j-1][j] = best
    else:
        F[i][j] = tspf(i, j-1, F, D) + D[j-1][j]
    return F[i][j]
```

Algorytmy zachłanne (ang. greedy)

- podejmuj decyzje, które "w tej chwili" wydają się najlepsze

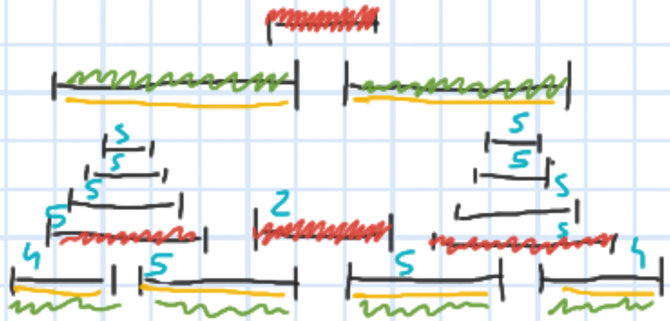
Problem wyboru zadań



Dane: zbiór przedziałów (zadań)

Zadanie: Wybrać jak najliczniejszy przedziałów, które nie mają części wspólnych

Pomysły



① "najkrótszy najpierw"

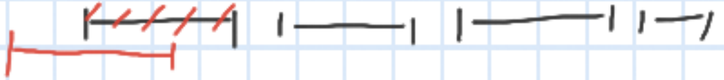
② "najmniej przecięć najpierw"

③ "najwcześniej kończący się najpierw"

Uzasadnienie poprawności

Rozważmy dowolne rozwiązanie optymalne

- jeśli zawiera najwcześniej kończący się przedział to OK
- jeśli nie zawiera to: dokładamy najwcześniej kończący się przedział i usuwamy to co przecina



Ciągły problem plecakowy

Dane: substancje $1, \ldots, n$

dla każdej substancji i mamy także

$v(i)$ – objętość i (dostępna)

$p(i)$ – wartość i

B – łączna objętość, którą możemy zabrać

Zadanie: zdecydować jaką objętość każdej substancji należy zabrać, żeby jej łączna wartość była maksymalna i nie przekroczyć B

① "najcenniejsza substancja najpierw"

$p(i)$	2	1	1	1	...	1
$v(i)$	B	1	1	1		1

② "najmniejsze najpierw"

$p(i)$	0.01	...	0.01	B	B	B
$v(i)$	1	...	1	2	2	2

③ Dla każdej substancji oblicz

$$\alpha(i) = \frac{p(i)}{v(i)}$$

wybieraj najbardziej opłacalne najpierw