Algorytmy zachłanne

Wybieranie lokalnie najlepszych działań w nadziei, że doprowadzi to do globalnie optymalnego rozwiązania. Odpowiednie problemy:

własność wyboru zachłannego: za pomocą lokalnie optymalnych (zachłannych) wyborów można uzyskać globalnie optymalne rozwiązanie

optymalna podstruktura: optymalne rozwiązanie jest funkcją optymalnych rozwiązań podproblemów

Problem wyboru zajęć

Dane: zbiór zajęć $S = \{1 \ldots n\}$, do których przydzielamy zasoby (n.p. sala). Dla $i = 1 \ldots n$, s_i – czas rozpoczęcia, f_i – czas zakończenia. Zadanie i zajmuje zasób przez przedział czasu $[s_i, f_i)$.

Wynik: Największy podzbiór nie zachodzących na siebie zajęć

Rozwiązanie – Wybór zachłanny: W każdym kroku: spośród tych zadań, które można jeszcze wybrać, dobieramy zadanie, które najwcześniej się kończy (pozostawia najwięcej swobody przy doborze pozostałych zadań)

Greedy-Activity-Selector

Zakładamy, że zajęcia są posortowane ze wzgl. na czas zakończenia:

$$f_1 \leq f_2 \leq \ldots \leq f_n$$

```
Greedy-Activity-Selector(s, f)
1 n ←length[s]
2 A ← {1}
3 j ← 1
4 for i ← 2 to n do
5    if s_i ≥ f_j then
6       A ← A ∪ {i}
7       j ← i
8 return A
```

Czas: $\Theta(n)$

Poprawność

Tw. Algorytm `Greedy-Activity-Selector` generuje rozwiązanie problemu wyboru zajęć o największym rozmiarze.

D-d. *Istnieje optymalne rozwiązanie, które zawiera* 1: Niech $A \subseteq S$ - optymalne rozwiązanie. Niech $k = \min A$. Jeśli $k > 1$, to $B = A \setminus \{k\} \cup \{1\}$ ma tyle samo elementów co A. Ponadto B jest rozwiązaniem, bo $f_1 \leq f_k$.

Optymalna podstruktura: Niech A – optymalne rozwiązanie dla S zawierające 1. Wtedy $A' = A \setminus \{1\}$ jest optymalnym rozwiązaniem dla $S' = \{i \in S : s_i \geq f_1\}$. (Gdyby istniało rozwiązanie B' dla S', takie że $|B'| > |A'|$, to zbiór $\{1\} \cup B'$ byłby większym od A rozwiązaniem dla S – sprzeczność, bo A – optymalny). $\square$

ciągły i dyskretny problem plecakowy

dyskretny: Dane n przedmotów; i-ty przedmiot wart c_i i waży w_i. Zapakować jak najcenniejszy ładunek do plecaka o nośności W.

ciągły: Można zabierać ułamkowe części przedmiotów ("substancji").

Wersja ciągła – algorytm zachłanny wg ceny jednostkowej: Tyle ile można najcenniejszej substancji, jeśli zostanie miejsce – tyle ile można kolejnej co do wartości, itd...

Wersja dyskretna – kontrprzykład: $W = 50$, $c_1 = 60$, $w_1 = 10$, $(c_1/w_1 = 6)$, $c_2 = 100$, $w_2 = 20$, $(c_2/w_2 = 5)$, $c_3 = 120$, $w_3 = 30$, $(c_3/w_3 = 4)$,

Strategia zachłanna daje: $\{1, 2\}$ ($w_1 + w_2 = 30$ przedmiot 3 się już nie zmieści). $c_1 + c_2 = 160$.

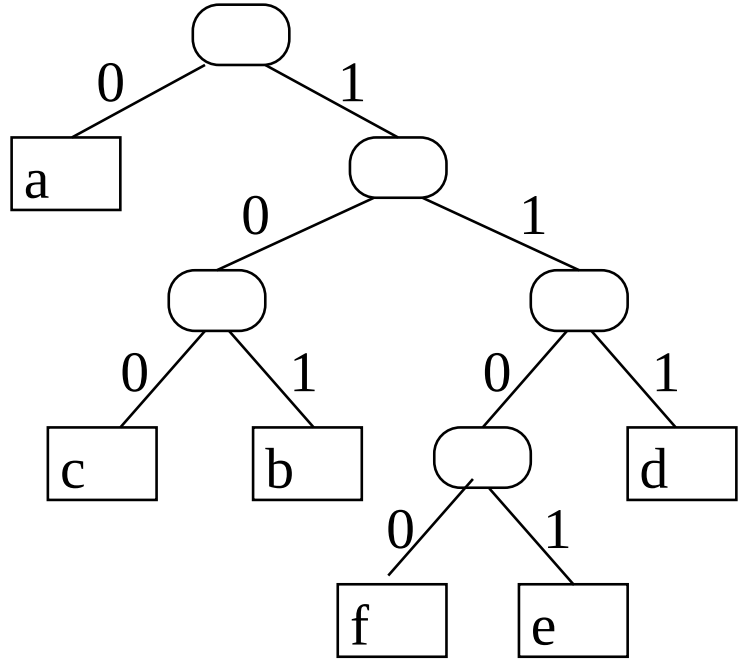
Optymalny wybór: $\{2, 3\}$ ($c_2 + c_3 = 220$).

Kody Huffmana

Kody prefiksowe – kod każdego znaku $c \in C$ jest ciągiem bitów, który nie jest prefiksem kodu innego znaku.
Wtedy: kod ciągu znaków – ciąg kodów znaków, może być jednoznacznie rozkodowany.

Drzewo T kodu prefiksowego

a − 0
b − 101
c − 100
d − 111
e − 1101
f − 1100



Długość zakodowanego tekstu: $B(T) = \sum_{c \in C} f(c) d_T(c)$,
gdzie $f(c)$ – liczba wystąpień c, $d_T(c)$ – długość kodu c.

Huffman

C - alfabet – zbiór znaków – liści drzewa. Każdy znak c ma atrybut – klucz $f[c]$ – liczba wystąpień.

```
Huffman(C)
1 n ← |C|
2 Q ← C
3 for i ← 1 to n − 1 do
4    z ← Allocate-Node()
5    x ← left[z] ←Extract-Min(Q)
6    y ← right[z] ←Extract-Min(Q)
7    f[z] ←f[x]+f[y]
8    Insert(Q, z)
9 return Extract-Min(Q)
```

Pętla `for` – wykonywana $|C| - 1$ razy. Jeśli kolejka priorytetowa Q – kopiec, to czas: $O(n \lg n)$.

Własność wyboru zachłannego

Lemat 1. Niech x i y – znaki o najmniejszej liczbie wystąpień. (Niech np. $f(x) \leq f(y)$.) Istnieje optymalny kod prefiksowy, w którym kody x i y różnią się tylko na ostatnim bicie.

D-d. Niech T – drzewo kodu optymalnego. Niech b i c – bliźniacze znaki – liście o największej głębokości w T, takie że $f(b) \leq f(c)$. Niech T' drzewo powstałe z T po zamianie pozycji x i b. Wtedy:

$$B(T) - B(T') = \sum_{c \in C} f(c) d_T(c) - \sum_{c \in C} f(c) d_{T'}(c)$$
$$= f(x) d_T(x) + f(b) d_T(b) - f(x) d_{T'}(x) - f(b) d_{T'}(b)$$
$$= f(x) d_T(x) + f(b) d_T(b) - f(x) d_T(b) - f(b) d_T(x)$$
$$= (f(b) - f(x))(d_T(b) - d_T(x)) \geq 0.$$

Niech T'' – drzewo T' po zamianie pozycji c i y. Wtedy $B(T') - B(T'') \geq 0$. Stąd $B(T'') \leq B(T)$, czyli T'' – optymalne. $\square$

Optymalna podstruktura

Lemat 2. Niech T – drzewo optymalne dla alfabetu C. Niech x i y – bliźniacze liście w drzewie T. Niech z – ojciec x i y w T. Wtedy, traktując z jako znak o liczbie wystąpień $f(z) = f(x) + f(y)$, $T' = T \setminus \{x, y\}$ reprezentuje kod optymalny dla alfabetu $C' = C \setminus \{x, y\} \cup \{z\}$.

D-d. Dla $c \in C \setminus \{x, y\}$ mamy $d_T(c) = d_{T'}(c)$, czyli $f(c)d_T(c) = f(c)d_{T'}(c)$. Ponieważ $d_T(x) = d_T(y) = d_{T'}(z) + 1$, zachodzi: $f(x)d_T(x) + f(y)d_T(y) = (f(x) + f(y))(d_{T'}(z) + 1)$ $= f(z)d_{T'}(z) + (f(x) + f(y))$. Stąd: $B(T) = B(T') + f(x) + f(y)$. Gdyby istniało T'' o liściach C', takie że $B(T'') < B(T')$, to podczepiając x i y pod z w T'' otrzymamy drzewo dla C o koszcie $B(T'') + f(x) + f(y) < B(T)$.(Sprzeczność.) □

Tw. `Huffman` generuje optymalny kod prefiksowy.

Matroidy

Matroid para $M = (S, \mathcal{N})$, taka że:

1. S jest skończonym zbiorem
2. $\mathcal{N}$ niepusta rodzina (*niezależnych*) podzbiorów S, taka że jeśli $B \in \mathcal{N}$ oraz $A \subseteq B$, to $A \in \mathcal{N}$ ($\mathcal{N}$ – *dziedziczne*)
3. Jeśli $A \in \mathcal{N}$, $B \in \mathcal{N}$ i $|A| < |B|$, to istnieje $x \in B \setminus A$, taki że $A \cup \{x\} \in \mathcal{N}$ (*własność wymiany*)

Przykład: *Matroidy macierzowe*
S – wiersze macierzy,
zbiór niezależny – liniowo niezależny podzbiór S.

Matroid grafowy

Dla grafu nieskierowanego $G = (V, E)$ definiujemy $M_G = (S_G, \mathcal{N}_G)$ następująco:

- $S_G = E$ – krawędzie G
- $A \in \mathcal{N}_G$ wtedy i tylko wtedy gdy A – acykliczny (las)

Tw. $M_G(S_G, \mathcal{N}_G)$ jest matroidem.

D-d. $S_G = E$ – skończony. $\mathcal{N}_G$ – dziedziczny (usunięcie krawędzi nie stworzy cyklu).
Własność wymiany:
Las, w którym jest 0 krawędzi ma $|V|$ drzew (izolowane wierzchołki). Dodanie do lasu krawędzi, która nie tworzy cyklu łączy dwa drzewa w jedno. Stąd las o k krawędziach ma $|V| - k$ drzew.

...

Matroidy

...

Niech $A, B \in \mathcal{N}_G$, $|A| < |B|$. Las B ma mniej drzew niż las A. Istn. drzewo T w lesie B, które zawiera wierzchołki z różnych drzew lasu A. Istnieje krawędź (u, v) w T, taka że u i v w różnych drzewach lasu A (bo T spójne). Można (u, v) dodać do A nie tworząc cyklu. $\square$

Matroidy

. . .

Niech $A, B \in \mathcal{N}_G$, $|A| < |B|$. Las B ma mniej drzew niż las A. Istn. drzewo T w lesie B, które zawiera wierzchołki z różnych drzew lasu A. Istnieje krawędź (u, v) w T, taka że u i v w różnych drzewach lasu A (bo T spójne). Można (u, v) dodać do A nie tworząc cyklu. $\square$

Dla matroidu $M = (S, \mathcal{N})$, x – *rozszerzenie* A jeśli $x \notin A$ i wierzchołki$A \cup \{x\} \in \mathcal{N}$.

A jest *maksymalny* jeśli nie ma rozszerzeń.

Tw. Wszystkie maksymalne podzbiory niezależne mają ten sam rozmiar.

D-d. Nie wprost: Niech $A, B \in \mathcal{N}$, A, B – maksymalne i $|A| < |B|$. Z własności wymiany istn. $x \in B \setminus A$, takie że $A \cup \{x\} \in \mathcal{N}$. Ale wtedy x – rozszerzenie A (sprzeczność). $\square$

Matroidy

Matroid *ważony* $M = (S, \mathcal{N})$, jeśli istnieje funkcja $w : S \to R^+$.
Rozszerzenie w na podzbiory S: $w(A) = \sum_{x \in A} w(x)$.
Optymalny – niezależny podzbiór S o największej wadze.
(wystarczy rozpatrywać maksymalne podzbiory, bo w – dodatnie.)

```
Greedy(M, w)
1  A ← ∅
2  posortuj S[M] nierosnąco wg w
3  for x ∈ S[M]  brane kolejno  do
4     if A ∪ {x} ∈ 𝒩[M]  then
5        A ← A ∪ {x}
6  return A
```

Czas: $O(n \lg n + nf(n))$, gdzie $O(n \lg n)$ – sortowanie, $O(f(n))$ – sprawdzenie $A \cup \{x\} \in \mathcal{N}[M]$

poprawność Greedy

Lemat 1. (własność zachłannego wyboru) Niech S – posortowany nierosnąco wg w. Niech x pierwszy w S, taki że $\{x\} \in \mathcal{N}$. Jeśli takie x istnieje, to istnieje optymalny A, taki że $x \in A$.

Jeśli takie x nie istnieje, to $\mathcal{N} = \{\emptyset\}$.

W p.p., niech B – optymalny, $B \neq \emptyset$. Jeśli $x \in B$, to wybieramy $A = B$.

W p.p. żaden element B nie ma wagi $> w(x)$: gdyby $y \in B$ i $w(y) > w(x)$, to $\{y\} \in \mathcal{N}$ (bo $\mathcal{N}$ – dziedziczne) – sprzeczność z wyborem x.

Konstrukcja A: Zaczynamy od $A = \{x\}$. Korzystając z własności wymiany dodajemy do A elementy z B aż $|A| = |B|$ zachowując $A \in \mathcal{N}$. Po zakończeniu $A = B \setminus \{y\} \cup \{x\}$ dla pewnego $y \in B$ i $w(A) = w(B) - w(y) + w(x) \geq w(B)$. Stąd A optymalny. $\square$

poprawność `Greedy`

Lemat 2. Jeśli $x \in S$ nie jest rozszerzeniem $\emptyset$, to x nie jest roszerzeniem żadnego niezależnego zbioru.

D-d. Nie wprost: Niech x – rozszerzenie A, i nie-rozszerzenie $\emptyset$. Wtedy $A \cup \{x\}$ niezależny. Z dziedziczności: $\{x\}$ – niezależny. Sprzeczność z tym, że x nie jest roszerzeniem $\emptyset$. $\square$

Wniosek: Elementy pominięte przez `Greedy` przed wybraniem pierwszego elementu są i tak "bezużyteczne".

poprawność Greedy

Lemat 3. (Optymalna podstruktura) Niech x – pierwszy wybrany element do zbioru. Niech $M' = (S', \mathcal{N}')$ (*kontrakcja*), taki że:

- $S' = \{y \in S \setminus \{x\} : \{x, y\} \in \mathcal{N}\}$
- $\mathcal{N}' = \{B \subseteq S \setminus \{x\} : B \cup \{x\} \in \mathcal{N}\}$
- waga M' – ograniczenie w do S'.

Wtedy, A' jest optymalny dla M' wtedy i tylko wtedy gdy $A' \cup \{x\}$ jest optymalny dla M.

Jeśli $A \in \mathcal{N}$, $x \in A$, to $A' = A \setminus \{x\} \in \mathcal{N}'$. I na odwrót: jeśli $A' \in \mathcal{N}'$ to $A = A' \cup \{x\} \in \mathcal{N}$. W obu przypadkach $w(A) = w(A') + w(x)$. Zatem aby wybrać "najcięższe" A zawierające x niezależne w M potrzeba i wystarczy wybrać najcięższe A' w M'. $\square$

poprawność `Greedy`

Tw. (Porawność `Greedy`) Wywołanie `Greedy(`M, w`)` wyznacza optymalny podzbiór.

Z lematu 2: każdy początkowo pominięty element może być pominięty.

Z lematu 1: pierwszy wybrany element należy do pewnego zbioru optymalnego dla M.

Z lematu 3: po wyborze pierwszego elementu x, problem polega na znalezieniu optymalnego podzbioru M'.

I tak się dzieje: Dalsze działanie `Greedy` można traktować jak wywołanie `Greedy` na M', (gdzie B – podzbiór niezależny, jeśli $B \cup \{x\} \in \mathcal{N}$, a więc B niezależny w M'), w efekcie którego wyznaczony zbiór jest postaci $\{x\} \cup A'$, gdzie A' – optymalny dla M'. $\square$

Szeregowanie zadań

Dane:

- $S = \{1 \ldots n\}$ – zbiór zadań o jednostkowym czasie wykonania
- ciąg: $d_1 \ldots d_n$, gdzie d_i termin dla i-tego zadania.
- ciąg: $w_1 \ldots w_n$, gdzie w_i – kara za przekroczenie terminu

Wynik: Uszeregowanie, w którym płacimy najmniejszą karę.
W uszeregowaniu: zadania *terminowe* i *spóźnione*.
Postać normalna – zadania terminowe poprzedzają spóźnione.
Postać kanoniczna – normalna, w której terminowe zadania posortowane rosnąco ze wzgl. na termin.
Każde uszeregowanie można sprowadzić do postaci kanonicznej nie zwiększając zbioru zadań spóźnionych.

Szeregowanie zadań

Podzbiór zadań A – *niezależny* jeśli można go uszeregować tak, że żadne zadanie A nie jest spóźnione.
Wniosek: wystarczy wyznaczyć niezależny zbiór zadań o największej sumie kar.
Dla zb. zadań A niech $N_t(A)$ – liczba zadań, których termin jest $\leq t$.

Lemat. Dla dowolnego zbioru zadań A następujące warunki są równoważne:

1. A jest niezależny
2. dla $t = 1 \ldots n$, $N_t(A) \leq t$
3. jeśli A posortowany niemalejąco wg terminu, to żadne zadanie nie jest spóźnione

Szeregowanie

D-d.

1.$\Rightarrow$2.: Jeśli $N_t(A) > t$ dla pewnego t, to nie jest możliwe terminowe wykonanie wszystkich zadań, bo więcej niż t trzeba wykonać przed terminem t.

2.$\Rightarrow$3.: Dla każdej chwili t zadań o terminie $\leq t$ jest $\leq t$ więc zdążą się wykonać w posortowanym uszeregowaniu.

3.$\Rightarrow$1.: z definicji. $\square$

Szeregowanie

D-d.
1.$\Rightarrow$2.: Jeśli $N_t(A) > t$ dla pewnego t, to nie jest możliwe terminowe wykonanie wszystkich zadań, bo więcej niż t trzeba wykonać przed terminem t.
2.$\Rightarrow$3.: Dla każdej chwili t zadań o terminie $\leq t$ jest $\leq t$ więc zdążą się wykonać w posortowanym uszeregowaniu.
3.$\Rightarrow$1.: z definicji. $\square$

Tw. Jeśli S jest zbiorem zadań a $\mathcal{N}$ jest rodziną niezależnych podzbiorów zadań, to $M = (S, \mathcal{N})$ jest matroidem.

Szeregowanie

D-d. Dziedziczność – oczywista.
Własność wymiany: Niech A,B – niezależne i $|B| > |A|$.
Niech $k = \max\{t : N_t(B) \leq N_t(A)\}$. Wiadomo, że $N_n(B) = |B|$ i $N_n(A) = |A|$ i $|B| > |A|$, więc $k < n$ oraz, dla $k < j \leq n$, zachodzi $N_j(B) > N_j(A)$. Stąd B zawiera więcej zadań o terminie $k+1$ niż A. Niech $x \in B \setminus A$ zadanie o terminie $k+1$. Niech $A' = A \cup \{x\}$. Z własności 2. lematu:
Dla $1 \leq t \leq k$ mamy $N_t(A') \leq N_t(A) \leq t$, bo A – niezależny.
Dla $k < t \leq n$ mamy
$N_t(A') = N_t(A) + N_t(\{x\}) \leq N_t(A) + 1 \leq N_t(B) \leq t$, bo B – niezależny.
Stąd A' – niezależny. □

Szeregowanie

D-d. Dziedziczność – oczywista.
Własność wymiany: Niech A,B – niezależne i $|B| > |A|$.
Niech $k = \max\{t : N_t(B) \leq N_t(A)\}$. Wiadomo, że $N_n(B) = |B|$ i $N_n(A) = |A|$ i $|B| > |A|$, więc $k < n$ oraz, dla $k < j \leq n$, zachodzi $N_j(B) > N_j(A)$. Stąd B zawiera więcej zadań o terminie $k+1$ niż A. Niech $x \in B \setminus A$ zadanie o terminie $k+1$. Niech $A' = A \cup \{x\}$. Z własności 2. lematu:
Dla $1 \leq t \leq k$ mamy $N_t(A') \leq N_t(A) \leq t$, bo A – niezależny.
Dla $k < t \leq n$ mamy
$N_t(A') = N_t(A) + N_t(\{x\}) \leq N_t(A) + 1 \leq N_t(B) \leq t$, bo B – niezależny.
Stąd A' – niezależny. $\square$
Wniosek: Można zastosować algorytm `Greedy`. Wtedy czas $O(n^2)$, bo sprawdzanie niezależności zajmuje $O(n)$ (korzystając z własności 2. lematu – ćw.).