1 Problem wydawania reszty

$c_1, c_2, \ldots, c_3 \in \mathbb{N}$ – nominały monet
$R \in \mathbb{N}$ – kwota do wydania

Zadanie:
Znaleźć wielozbiór S elementów $c_1, c_2, \ldots, c_n$ taki, że $\sum_{x \in S} x = R$ i $|S|$ – minimalne.
Przykład:
Nominały: $1, 2, 5, 10, 20$
$R = 75 = 20 + 20 + 20 + 10 + 5$
$S = \{20, 20, 20, 10, 5\}$, $|S| = 5$
Jest to postępowanie zachłanne: dobieramy największy nominał dopóki możemy.

UWAGA! Postępowanie zachłanne może nie znajdować rozwiązania – przykład:
$R = 100$, $c_1 = 50, c_2 = 51$

2 MST

Zadanie:
Znaleźć minimalne drzewo rozpinające grafu. Drzewem rozpinającym jest taki podzbiór krawędzi, że istnieje ścieżka pomiędzy dowolnymi dwoma wierzchołkami grafu. Minimalnym drzewem jest natomiast takie drzewo, dla którego suma wag krawędzi będzie minimalna spośród wszystkich drzew rozpinających danego grafu.

Znane algorytmy:

- algorytm Kruskala
- algorytm Prima
- algorytm Boruvki

2.1 Kruskal

$e_1, e_2, \ldots e_m$ – krawędzie posortowane długością
$E = \emptyset$
Do E po kolei dorzucamy krawędzie $e_1, e_2, \ldots e_m$, które nie kolidują z poprzednimi (przez kolidują rozumiemy, że powstaje cykl). Na koniec E jest poszukiwanym zbiorem krawędzi.

2.2 Boruvki

1. dla każdego wierzchołka znajdujemy najtańszą krawędź
2. tworzymy nowy graf taki, że stare spójne składowe stają się wierzchołkami, a krawędzie między spójnymi stają się krawędziami

Powyższe kroki wykonujemy dopóki nie pozostanie jedna spójna składowa.

Istotną cechą tego algorytmu jest to, że można zrównoleglić obliczenia.

3 Set cover

Dane:
$\mathbb{U}$ – uniwersum, $|\mathbb{U}| = n$, $\mathbb{S} = \{S_1, \ldots, S_k\}$,
$\forall_{S \in \mathbb{S}} S \subset \mathbb{U}$ oraz $\cup_{S \in \mathbb{S}} S = \mathbb{U}$, $c : \mathbb{S} \to R_+$
Zadanie:
Znaleźć $S' \subset \mathbb{S}$ takie, że $\cup_{x \in S'} x = \mathbb{U}$ takie, że $koszt(rozw) = \sum_{x \in S'} c(x)$ – minimalny

Dla tego przykładu nie istnieje rozwiązanie zachłanne. Można jednak (stosując algorytm zachłanny) uzyskać rozwiązanie niedużo gorsze od optymalnego.

Startegia:

W każdym kroku do S' dobieramy zbiór taki zbiór X, który ma najmniejszą $kosztownosc = \frac{c(X)}{c(X \setminus \cup_{X \in S'} X)}$

Fakt Koszt rozwiązania otrzymanego tym algorytmem $koszt(rozw) \leq O(\log n) \cdot Opt$ (Opt - koszt rozwiązania optymalnego)

Dowód

Niech $e_1, e_2, \ldots, e_n$ - ciąg elementów $\mathbb{U}$ w kolejności pokrywania przez algorytm.

$koszt(rozw) = \displaystyle\sum_{i=1}^{n} cena(e_i)$

$cena(e_i)$ - średnia cena za element e_i w momencie brania zbioru do którego należy

W momencie pokrywania elementu e_i liczba niepokrytych elementów $\geq n - i + 1$. Gdyby teraz algorytm ograniczył się do brania zbiorów z rozwiązania optymalnego, średnia cena tych elementów $\leq \frac{Opt}{n-i+1}$.

Wniosek: $cena(e_i) \leq \frac{Opt}{n-i+1}$, więc $koszt(rozw) \leq \displaystyle\sum_{i=1}^{n} \frac{Opt}{n-i+1} = Opt \cdot \sum_{i=1}^{n} \frac{1}{i} \leq Opt \cdot \log n$