Algorytmika - LAB

Algorytmy to serce informatyki.
Wiedza o nich daje niezliczone zastosowania praktyczne.
Poznawanie ich pozwala na przygodę intelektualną.

Algorytm to zestaw dobrze zdefiniowanych instrukcji, które należy wykonać, aby rozwiązać określony problem.

To, co decyduje o jakości algorytmu to jego poprawność i wydajność (złożoność)

Mamy wypracowane różne strategie projektowania algorytmów takie jak na przykład: wyczerpująca, zachłanna , dynamiczna czy rekurencyjna

Mamy wypracowany sposób oceny złożoności algorytmów: analiza asymptotyczna

Poprawność algorytmu

Problem kasjera

wydać resztę optymalnie czyli
wykorzystując możliwie najmniejszą liczbę monet przy zadanym zestawie monet.

Wydawaj możliwie największą monetę z aktualnej reszty

```
USCHANGE(M)
1  while M > 0
2      c ← Largest coin that is smaller than (or equal to) M
3      Give coin with denomination c to customer
4      M ← M − c
```

Problem kasjera: strategia zachłanna

Strategia *zachłanna*: wydawaj resztę aktualnie największym nominałem

```
BETTERCHANGE(M, c, d)
1  r ← M
2  for k ← 1 to d
3        i_k ← r/c_k
4        r ← r − c_k · i_k
5  return (i_1, i_2, ..., i_d)
```

Ile tych możliwie najwiekszych monet można wypłacić

Algorytm BetterChange jest niepoprawny.
Źle oblicza resztę dla np.:
M=40, c=(25 , 20, 10, 5, 1)

Problem kasjera : strategia wyczerpująca

Strategia *wyczerpująca*: kolejno rozważ wszystkie możliwe kombinacje monet

Algorytm zawsze poprawny, ale też i zawsze niewydajny

Algorytm mięśniaka

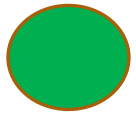
```
BRUTEFORCECHANGE(M, c, d)
1  smallestNumberOfCoins ← ∞
2  for each (i_1, ..., i_d) from (0, ..., 0) to (M/c_1, ..., M/c_d)
3      valueOfCoins ← Σ_{k=1}^{d} i_k c_k
4      if valueOfCoins = M
5          numberOfCoins ← Σ_{k=1}^{d} i_k
6          if numberOfCoins < smallestNumberOfCoins
7              smallestNumberOfCoins ← numberOfCoins
8              bestChange ← (i_1, i_2, ..., i_d)
9  return (bestChange)
```

Złożoność algorytmu

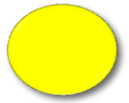
Analiza wydajności algorytmu

Analiza wydajności algorytmu to ocena czasu przetwarzania

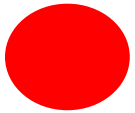
ile umownych jednostek czasowych potrzeba na przetworzenie n-elementowego zbioru:



analiza najgorszego przypadku:
T(n)=max { czas sortowania danych o rozmiarze n}



analiza średniego przypadku:
T(n)= E { wszystkie możliwe dane o rozmiarze n}



analiza najlepszego przypadku:
możliwość oszukania się i innych

Analiza asymptotyczna

Pomysł na ilościową ocenę wydajności algorytmu

ANALIZA ASYMPTOTYCZNA

1. zignoruj stałe wielkości zależne od maszyny
2. obserwuj jedynie **WZROST** przebiegu **T(n)**, gdy **n** → ∝

Zadanie na rozgrzewkę

Zaimplementuj obie funkcje.

Przeprowadź pomiar czasu wykonania przy różnych zestawach monet.