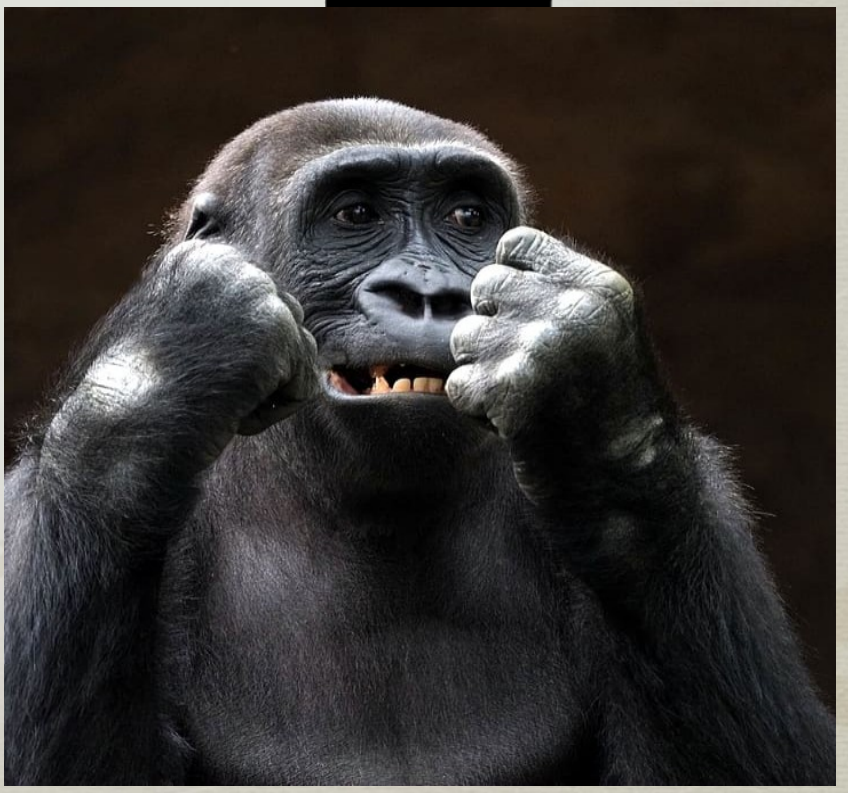
Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Groba sila in izčrpno preiskovanje

Jurij Mihelič, UniLj, FRI

Groba sila



Groba sila

- Ideja metode
  - direkten pristop glede na definicijo problema
  - pregledovanje vseh možnosti
  - brez ubiranja bližnjic in optimizacije
  - algoritmi so preprosti in redko optimalni
- Uporaba
  - nimamo druge ideje
  - hitro rabimo enostavno rešitev
  - za preproste probleme
  - za majhne naloge

Množenje in potenciranje

- Množenje
  - zaporedno prištevanje
  - $a \cdot b = a + a + \ldots + a$ … b-1 seštevanj
- Potenciranje
  - zaporedno množenje
  - $a^b = a \cdot a \cdot \ldots \cdot a$ … b-1 množenj

Ovrednotenje polinoma

- Problem
  - $p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x^1 + a_0$

    $= (\ldots((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \cdot x \ \ldots + a_1) \cdot x^1 + a_0$

- Razvoj algoritma

```
px = a[0]
for i = 1 to n do
    xn = x
    for j = 1 to i-1 do
        xn = xn * x
    px = px + a[i] * xn
```

n seštevanj
n(n+1)/2 množenj

```
px = a[0]
xn = x
for i = 1 to n do
    px = px + a[i] * xn
    xn = xn * x
```

n seštevanj
2n množenj

Hornerjev algoritem

```
px = a[n]
for i = n-1 downto 0 do
    px = px * x + a[i]
```

n seštevanj
n množenj

Iskanje minimuma

- Problem

Iskanje minimuma
- naloga: seznam $L = [x_1, x_2, \ldots, x_n]$ števil
- rešitev: število $m \in L$, kjer $m \leq x$ za vsak $x \in L$

- Razvoj algoritma
  - opis algoritma
  - pravilnost algoritma

Iskanje najmanjšega števila v seznamu števil (minimum)
Vsak element seznama primerjaš s prvim in če je element manjši ju zamenjaš. Prvi element tako postane najmanjši.

- Analiza algoritma
  - časovna zahtevnost
  - prostorska zahtevnost

```
for i = 0 to n-1 do
    if L[i] < L[0] then
        swap(L, i, 0)
return L[0]
```

Iskanje elementa

- Problem

Iskanje elementa
- naloga: seznam $L = [x_1, x_2, \ldots, x_n]$ in element x
- rešitev: true, če $x \in L$, sicer false

- Razvoj algoritma
  - zaporedoma preveri vse elemente seznama

```
for i = 0 to n-1 do
    if L[i] == x then
        return true
return false
```

- Analiza algoritma
  - št. primerjav
  - čas izvajanja

Napovedovanje časa izvajanja

- Postopek
  - časovna zahtevnost
    - linearna enačba

$$T(n) = a \cdot n + b$$

  - določimo $T(n_1)$ in $T(n_{12})$
    - naredimo praktični preizkus pri različnih n_1 in n_2

$$T(n_1) = a \cdot n_1 + b$$
$$T(n_2) = a \cdot n_2 + b$$

  - izračunamo a in b
    - odštejemo enačbi

$$a = \frac{T(n_2) - T(n_1)}{n_2 - n_1}$$

$$b = T(n_2) - a \cdot n_2$$

Iskanje podniza

- Problem
  - v danem besedilu T (*text*) poišči podniz P (*pattern*)

Iskanje podniza
- naloga: niza T in P
- rešitev: true, če se P nahaja v T; sicer false

- Razvoj algoritma
  - preveri vse pozicije v T, če je tam P

- Analiza algoritma
  - $n = |T|, m=|P|$
  - $m \leq n$, torej $m=O(n)$

```
fun check(T, P, i) is
  for j = 0 to m-1 do
    if T[i + j] == P[j] then
      return true
  return false

fun find(T, P) is
  for i = 0 to n - m do
    if check(T, P, i) then
      return true
  return false
```

Izčrpno preiskovanje

Izčrpno preiskovanje

- Ideja metode
  - podvrsta grobe sile, včasih metodi enačimo
  - sistematično preverjanje vseh dopustnih rešitev
    - vsako rešitev preverimo, če ustreza glede na definicijo
    - izberemo najbolj ustrezno
- Uporaba
  - reševanje kombinatoričnih problemov
    - podmnožice, kombinacije, permutacije, …
  - za manjše naloge

Par najbližjih točk

- Problem
  - v danem seznamu točk poišči najbližji dve
- Razvoj algoritma
  - iskanje minimuma razdalje med vsemi možnimi pari točk
- Analiza algoritma
  - n … število točk
  - $T(n)=O(n^2)$

Največji podseznam

- Problem
  - v danem seznamu števil poišči (strnjen) podseznam z največjo vsoto
- Razvoj algoritma
  - preverimo vse možne podsezname
- Analiza algoritma
  - n … velikost seznama
  - $T(n)=O(n^3)$

Trgovski potnik

- Problem
  - v omrežju poišči najcenejši Hamiltonov obhod
- Ideja izčrpnega preiskovanja
  - generiraj vsa možna zaporedja vozlišč
  - preveri, če gre za Hamiltonov obhod
  - izberi najcenejši obhod
- Zahtevnost
  - št. zaporedij: $O(n!)$

Ostali primeri

- Uporaba grobe sile in izčrpnega preiskovanja
  - urejanje z izbiranjem
  - urejanje z mehurčki
  - iskanje duplikatov (vsak z vsakim)
  - Hamiltonov obhod, trgovski potnik
  - klika, neodvisna množica
  - vozliščno pokritje, dominantna množica
  - 0/1 nahrbtnik
  - itd.