Anotace

- Rychlé mocnění
- Fronta a zásobník
- Síla předvýpočtu
- Rekurze podruhé

Rychlé mocnění

- Mocnit (a^b) lze mnoha zpusoby.
- Např.: $a^b = a * a^{b-1}$
- Pak je složitost lineární v b.
- Nebo exponent vyjádříme ve dvojkové soustavě a pak:
- $x^{\alpha_1 \ldots \alpha_k 0} = x^{\alpha_1 \ldots \alpha_k 2}$ a $x^{\alpha_1 \ldots \alpha_k 1} = x^{\alpha_1 \ldots \alpha_k 2} * x$
- A složitost je lineární vuči dvojkovému zápisu exponentu!

Fronta a zásobník

- Fronta je datová struktura osazená operacemi `zařaď` a `vyřaď`,
- `zařaď` zařadí daný prvek na konec fronty,
- `vyřaď` vyřadí daný prvek ze začátku fronty.
- Zásobník je datová struktura osazená operacemi `push` a `pull`.
- `push` přidá na konec zásobníku, `pull` vytáhne z konce zásobníku.

Figurky na šachovnici anebo prohledávání grafu - algoritmus vlny

- Na děravé šachovnici je umístěn král. Určete jak rychle se dostane na určené (cílové) pole?
- Vlastně jde o úlohu prohledávání grafu.
- Podobné: Thesseus hledá Mínótaura.
- Rozdíl: Tehdy nám nešlo o nejkratší cestu.

Algoritmus vlny (myšlenky)

- Na šachovnici vysadíme na specifikované políčko mravence,
- mravenci polezou rovnoměrně všemi dostupnými směry,
- mravenci zkusí všechny cesty, rychleji než první mravenec se tam dostat nelze.
- Jiná intuice: Na příslušné poličko vychrstneme vodu,
- voda teče všemi dostupnými cestami, až doteče na cílové políčko.

Algoritmus vlny (implementace)

- Vytvoř prázdnou frontu f.
- Nastav vzdálenost do všech políček kromě startovního na nekonečno, vzdálenost do startovního nastav na 0.
- Přidej do f startovní políčko.
- Dokud není fronta f prázdná, opakuj:
  - a:=vyřaď(f);
  - Pro všechny sousedy z pole a zkus:
  - Pokud vzdálenost do z je větší než vzdálenost do a +1, nastav políčku z vzdálenost $a+1$ a zařaď(z, f).

Maximální jedničková podmatice

Problém: V matici tvaru $m \times n$ vyplněné nulami a jedničkami máme najít největší podmatici (souvislou) obsahující pouze jedničky.

Naivní algoritmus

- Pro každý možný levý horní a pravý dolní roh prohlédni vnitřek matice.
- Algoritmus funguje, ale s jakou složitostí?
- $\Theta(mn)$ levých horních rohů, $\Theta(mn)$ pravých dolních rohů, $\Theta(mn)$ prvků uvnitř (proč?), celkem tedy $\Theta(m^3n^3)$.
- Nápady na zlepšení?

Předvýpočet

- Pro každou jedničku si spočítáme, kolik jedniček leží bezprostředně pod ní (tedy v řadě nepřerušené nulou).
- Zkoušíme každou matici identifikovat pomocí levého a pravého horního rohu:
  - Ke kandidátu na levý horní roh zkoušej všechny možnosti pravého horního rohu (v jeho řadě).
  - Tyto nesmějí být odděleny nulou (tedy "žijí" v souvislém bloku jedniček),
  - Jelikož máme posčítané počty jedniček směrem dolů, stačí jako druhý rozměr vzít minimum z těchto posčítaných jedniček.
  - Zbytek je násobení a porovnávání.
- Složitost: Předvýpočet $O(mn)$, výpočet $O(m^2 n)$.

Lze to ještě urychlit?

Ku podivu ano – a ještě k tomu znovu předvýpočtem:

- Spočítej blok jedniček směrem dolů, ($\rightarrow B$)
- spočítej blok jedniček směrem nahoru, ($\rightarrow C$)
- zkusíme hledat "levý kritický konec", tedy místo, kde matice "najede na nulu", tedy $a_{i,j} = 1$ a $a_{i,j-1} = 0$ nebo $j = 1$ ($a_{i,j-1}$ leží mimo matici).
- Zkus všechny možné kandidáty na pravý okraj (v příslušném řádku).

Analýza složitosti

- Předvýpočty (stavba matic B a C): $O(mn)$,
- ačkoliv se zdá, že složitost výpočtu bude stejná jako dříve, není tomu tak,
- protože každý prvek matice zkoušíme jako kandidát na "pravý okraj" jen jednou!
- Celkem tedy $O(mn)$; jelikož složitost problému je $\Omega(mn)$, máme algoritmus (až na konstantu) optimální.

Rekurze podruhé

- Rekurze je metoda řešení problému spočívající v tom, že problém "zmenšíme" a z řešení menší instance odvodíme řešení větší instance.
- Příklady: faktoriál,
- Dnes: Výpis všech čísel v zadané číselné soustavě (o dané délce),
- Přednášející jde do F1.

Hlavní program

```
program q;
const MAX=10;
var cif,zakl,pocet:integer;
      pole:array[1..MAX] of integer;
begin
      write('Zadej pocet cifer: ');
      readln(cif);
      if(cif>MAX) then
            halt;{Moc dlouhe cislo}
      write('Zadej zaklad soustavy: ');
      readln(zakl);
      if zakl>10 then
            halt;{moc vysoky zaklad soustavy}
      vypln(1);
end
```

Jádro rekurze

```
procedure vypln(odkud:integer);
var i:integer;
begin
      if(odkud<=cif) then
            for i:=0 to zakl-1 do
            begin
                  pole[odkud]:=i;
                  vypln(odkud+1);
            end
      else vypis;
end;
```

Procedura vypis

```
procedure vypis;
var i:integer;
start:boolean;
begin
      start:=true;
      for i:=1 to cif do
            if((not start) or (pole[i]<>0)) then
            begin
                  start:=false;
                  write(pole[i]);
            end;
      if start then write(0);
      writeln;
end;
```

Přednášející jde do F1

- Přednášející při cestě po posluchárny na schodech vždycky buďto šlápne na následující schod, nebo jeden schod překročí.
- Kolika způsoby může vylézt do posluchárny F1?
  (schody nepočítejte, jejich počet odhadněte)
- Nápady?

Přednášející do posluchárny – řešení

- Vzniká rekurence $f_n = f_{n-1} + f_{n-2}$.
- Rekurence není nic jiného, než matematický zápis rekurze.
- Řešení:

```
function schody(a:integer):integer;
begin
    if a=1 then schody=1
    else if a=2 then schody=2
        else
            schody:=schody(a-1)+schody(a-2);
end;
```

- Jaký je problém?
- Ano, příšerná složitost.