

Anotace

- Síla předvýpočtu
- Rekurze podruhé

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 zkusíš 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 $O(m^2n)$.

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 zkusíme 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, přednášející jde do M1...
- Dnes: Výpis všech čísel v zadané číselné soustavě (o dané délce),
- Problém batohu

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;
```

Problém batohu

- Vykrademe klenotnictví a chceme si odnést co největší lup – krademe jen zlato, tedy hmotnost odpovídá ceně.
- Problém je těžký (NP-úplný) pro neomezené hmotnosti,
- pokud se hmotnosti jednotlivých předmětů dostatečně liší, lze problém řešit polynomiálně,
- tedy například jsou-li hmotnosti celočíselné.

Problém batohu

- Jak vyřešit rekurzí?
- Budeme vždycky zkoušet: Buď to prvek přidáme, nebo ne.
- Tedy načteme vstup (do pole), začneme od první položky a každou postupně zkusíme:
 - 1 přidat,
 - 2 nepřidat.

Hlavní program

```
program knapsack;
const MAX=10;
var kap,pocet,i:integer;
    polozky:array[0..MAX] of integer;
    pridano:array[0..MAX] of boolean;
begin {nacteni}
    readln(kap); readln(pocet);
    polozky[0]:=0;
    for i:=1 to pocet do
        begin
            readln(polozky[i]);
            pridano[i]:=false;
        end;
    pridej(0);
end
```

```
procedure pridej(odkud:integer);
```

```
var i:integer;
begin if kap>0 then
    for i:=odkud+1 to pocet do
    begin pridano[i]:=true;
        kap:=kap-polozky[i];
        pridej(i);
        kap:=kap+polozky[i];
        pridano[i]:=false;
    end
else if kap=0 then
    begin for i:=1 to MAX do
        if pridano[i] then write(i,', ');
        writeln;
    end;
end;
```