Anotace

- Dynamické programování,
- Grafové algoritmy.

Problémy řešitelné vyplněním tabulky

- Faktoriál,
- přednášející šel do F2,
- počet platných uzávorkování pomocí n párů závorek,
- Pascalův trojúhelník,
- počet rozkladů přirozeného čísla na součet nerostoucích kladných celých čísel,
- závorkování matic (pro účely násobení),
- nejdelší rostoucí podposloupnost,
- problém batohu.
- Grafové algoritmy.

Nejdelší rostoucí podposloupnost

- Utvoř posloupnost dvojic (třeba pomocí pole recordů),
- první prvek obsahuje příslušnou hodnotu, druhý ukazuje délku nejdelší rostoucí podposloupnosti končící dotyčným prvkem.
- Vyplňuj zleva doprava, pro každý prvek najdi mezi prvky jemu předcházejícími takový menší prvek, ve kterém končí nejdelší dostupná podposloupnost.
- Podposloupnost najdi od konce:
- Najdi prvek nabízející největší možnou délku,
- poznamenej si HODNOTU a DÉLKU.
- postupuj od konce a pokud najedeš na prvek nabízející délku DÉLKA s hodnotou nejvýše tolik, kolik HODNOTA, prvek si poznamenej, sniž DÉLKU o jedna a HODNOTU na hodnotu nalezeného prvku.
- Nalezenou posloupnost otoč.

Nejdelší rostoucí podposloupnost – výpočet (vyplnění tabulky)

```
for i:=1 to n do begin
     maximum:=0; maxindex:=0;
     for j:=i-1 downto 1 do begin
          if pole[j].hodnota<pole[i].hodnota
                         and pole[j].delky>maximum then
          begin
               maximum:=pole[j].delky;
               maxindex:=j;
          end;
     pole[i].delky:=maximum+1;
end;
```

Problém batohu

Definition

Problémem batohu nazveme problém, kdy máme zadány váhy jednotlivých předmětů $(w_1, \ldots w_n)$ a jejich ceny $(c_1, \ldots c_n)$ a nosnost batohu m a ptáme se, jak naložit batoh co nejcennějším obsahem.

- Předpokládejme variantu, že všechna čísla jsou přirozená!
- Rekurzivní řešení:
- Načti předměty a volej funkci přidej(k,l).
- První parametr říká, kolik předmětů už v batohu je, druhý určuje index posledního z nich.
- Nevýhoda: Zkoušíme všechny možnosti.
- Zlepšení? Dynamickým programem, ale neotřelým způsobem.

- Vytvoř pomocné batohy s nosnostmi $1, 2, \ldots m$ a proměnné popisující optimum v nich zinicializuj nulami.
- Pro každý předmět i přepočítej všechny batohy a zjisti, zda dopadnou lépe, pokud předmět přidáme nebo ne:
- Pro batoh k zjisti, pokud `cena_batohu(`k`)<cena_batohu(`$k - w_i$`)` $+ c_i$.
- Pokud ano, `cena_batohu(`k`):=cena_batohu(`$k - w_i$`)` $+ c_i$.
- Invariant: Řešíme-li prvek i, evidujeme (před touto fází) optimální naplnění batohů pomocí prvků $1, 2, \ldots k - 1$.
- Složitost: mn.
- Pozor, pokud nejsou čísla přirozená, stává se problém NP-těžkým. Není ale silně NP-těžký, což znamená, že těžkost se vynucuje "velkými čísly lišícími se o málo".

Grafově-optimalizační problémy

- Jak reprezentovat grafy – bylo v zimě, nyní už snad dokážete i implementovat.
- Jak grafy prohledávat – také bylo (do šířky a do hloubky), nyní umíte též implementovat.
- Hledání nejkratší cesty, minimální kostra,
- topologické uspořádání, faktorová množina (vyšetřování komponent).
- Modifikace problémů: Maximální kostra, nejdelší cesta – čím se liší?

Nejkratší cesta

- **Dijkstrův algoritmus:** Modifikované prohledávání do šířky (netvoříme frontu nalezených vrcholů, ale strukturu organizujeme podle doby, kdy se do dotyčného vrcholu dostaneme).
- Povšimněte si, že se vlastně jedná o diskrétní simulaci (rozlijeme vodu na graf a čekáme, kdy voda doteče do jednotlivých vrcholů).
- **Bellman-Fordův algoritmus:** $n - 1$-krát zopakujeme: Z každého vrcholu zkus "natáhnout" cestu po všech hranách z něj vycházejících (tedy zlepši případnou nalezenou cestu).
- Algoritmus funguje i při záporném ohodnocení hran, nesmí ale být přítomna záporná kružnice (kružnice záporné délky).

Floyd – Warshallův algoritmus

aneb all-pairs shortest paths

- Další příklad dynamického programování!
- Pro trojici (a, u, v), kde u, v jsou vrcholy a a přirozené číslo počítáme nejkratší cestu z u do v o nejvýše a hranách.
- Postupujeme pro rostoucí a (projdeme všechny možné dvojice) tak, že natahujeme cestu o jedna kratší o "poslední" hranu.
- Jako by vybízelo k rekurzi...
- ... jenže jako obvykle velmi neefektivní.
- Tedy ji vybavíme cachí v podobě třírozměrného pole...
- ... a zjistíme, že rekurzi vůbec nepotřebujeme, že postačí čtyři cykly v sobě...

Floyd – Warshallův algoritmus pseudokód

```
for a:=1 to n-1 do
   for u in vrcholy do
      for v in vrcholy do
         for w in sousedi(v) do
            cache[a,u,v]:=min(cache[a,u,v],cache[a-1,u,w]+
                 length(w,v));
```

Minimální kostra

- Vstup: Graf s ohodnocenými hranami
- Cíl: Kostra s minimální váhou.
- Algoritmy: Kruskal, Borůvka, Jarník, Prim.
- Kruskalův: Setřiď hrany podle váhy, postupně zkoušej přidávat a koukej, zda jsme vytvořili kružnici (pokud ano, hranu nepřidej, jinak přidej).
- Borůvkův: Spojování komponent souvislosti: Vyber hranu s nejmenší vahou, která vychází z dané komponenty a přidej.
- Jarníkův (Primův): Pěstování stromu: K dosud postavenému stromu přidej hranu z něj vycházející, která má nejnižší váhu.
- Všechny algoritmy počítají to samé. Důkazy korektnosti budou příště.

Modifikace

- Hledání nejtěžší kostry...
- ... změň na každé hraně váhu z v_i na $-v_i$.
- Hledání nejdelší cesty...
- ... těžké (NP-těžké).
- Proč? Protože víme, kolik hran má kostra, ale nevíme, kolik hran má cesta.
- Tudíž i nalezení nejkratší cesty v (potenciálně záporně) ohodnoceném grafu je těžké (NP-těžký problém):
- Sedí za ním schovaná Hamiltonskost, tedy hledání cesty délky $n - 1$: