Anotace

- Středník II!! 7. 5. 2010
- Odstranění rekurze,
- binární soubory,
- programování her.

Odstranění rekurze obecně

- Rekurze je pěkná věc, ale podle teorie lze každý program napsat v podobě jednoho jediného cyklu (while) bez volání dalších funkcí.
- Sice takový program není k přečtení, ale jedná se o zajímavý teoretický závěr.
- Jak to udělat?
- Uděláme totéž, co za nás udělá překladač, když spustíme rekurzi, tedy...

Odstranění rekurze

- Vyrobíme nové lokální proměnné, skočíme na správné místo programu a zapamatujeme si, kam se máme vrátit.
- Ve skutečnosti si jen (na zásobník) uložíme údaje o dosavadních datech, nahradíme daty "zarekurzenými" a údaj odkud jsme se "zavolali" a skočíme na "začátek" funkce.
- Při "odrekurzení" poznamenáme návratové údaje, vytáhneme ze zásobníku původní obsahy lokálních proměnných a údaj odkud jsme se zavolali (a skočíme tam).

Quicksort s logaritmickou pamětí

- Quicksort může vytvořit velkou a malou část.
- Pak potřebujeme lineárně paměti při klasickém rekurzívním řešení.
- Při třídění druhé části nepotřebujeme návratové údaje.
- Proto napřed setřídíme část, která je menší (co do množství dat).
- A problém můžeme řešit hybridně, tedy na první část se rekurzívně zavolat (i když i v tomto případě umíme rekurzi odbourat, neumíme odbourat její paměťové nároky).
- Tím, že rekurzi (nebo její náhražku) spustíme jen na menší část, pracujeme v i-té úrovni rekurze nejvýše s $\frac{n}{2^i}$ hodnotami.

Quicksort s omezenou rekurzí

Hybridní implementace v pseudokódu

```
procedure quicksort(levy,pravy);
begin
      while (levy<>pravy) do
      begin index_pivota:=rozdel;
            if(index_pivota>(pravy-levy)/2) then
            begin quicksort(index_pivota+1,pravy);
                  pravy:=levy+index_pivota
            end else begin
                  quicksort(levy,index_pivota);
                  levy:=levy+index_pivota+1;
end;  end;  end;
```

Textové a binární soubory

Zhodnocení situace

- V zimě jsme se učili pracovat s textovými soubory.
- Proměnnou typu `text` jsme napojili na soubor, otevřeli jsme, četli, zapisovali a zavřeli.
- Široká použitelnost, občas ale chceme naprogramovat například databázi (knih v knihovně).
- K tomu by se hodil soubor úplně jiných vlastností, ve kterém půjde lépe vyhledávat.
- K tomu přesně lze použít binární soubory.
- Jedná se o soubor sestávající z prvků popsané struktury, které můžeme číst resp. zapisovat.
- Jelikož známe velikost dotyčné struktury, můžeme i vyhledávat.

Technické prostředky

- Většinou se shodují s textovými soubory, nicméně:
- Uděláme proměnnou typu `file of nacitany_typ`.
  Například: `var f:file of nesmysl;`
- Funkce `assign`, `reset`, `rewrite`, `read`, `write` a `close` fungují úplně stejně (tedy jako první argument jim předáme příslušnou proměnnou typu `file`.
- Pozor na `append`!
- Hlavní rozdíl:
  - `filesize` – zjistí počet záznamů v souboru,
  - `seek` – nastaví ukazatel na příslušné místo.

Příklad

telefonní seznam

```
type zaznam=record
       jmeno:string[100];
       cislo:string[20];
end;
var f:file of zaznam;
```

Příklad přidání

do binárního souboru

```
procedure pridej;
var zazn:zaznam;
begin readln(zazn.jmeno);
      readln(zazn.cislo);
      assign(f,'databaze.bin');
      {$I-}
      reset(f);
      {$I+}
      if IOResult<>0 then
            rewrite(F);
      seek(f,filesize(f));
      write(f,zazn);
      close(f);
end;
```

Výpis souboru

```
procedure vypis;
var i:integer;
    zazn:zaznam;
begin
      assign(f,'databaze.bin');
      reset(f);
      for i:=1 to filesize(f) do
      begin
            read(f,zazn);
            writeln('Jmeno: ',zazn.jmeno,', cislo:
',zazn.cislo);
      end;
      close(f);
end;
```

Mazází v binárním souboru

Funkce `truncate`

- Zbytek binárního souboru můžeme smazat pomocí funkce `truncate`.
- Jako parametr jí předáme proměnnou typu file (napojenou na nějaký soubor).
- Příklad:
  ```
  reset(f);
  truncate(f);
  ```
  smaže dosavadní obsah souboru
- podobně jako by udělalo `rewrite(f)`, ovšem pro neexistující soubor by to nový nezaložilo!
- Jak zrušit jeden záznam?
- Přepsat ho posledním a poslední zrušit.

Teorie her

- Kombinatorická hra je hrou dvou hráčů. Stav hry je určen pozicí nějakých předmětů. Všechny zúčastněné předměty jsou viditelné. Jde o tzv. hru s úplnou informací.
- Příklad: Nimm, Podivná hra, Dáma, Šachy, Halma, Mlýn, Otrávená čokoláda...
- Kombinatorickými hrami nejsou: Poker, Prší, Mariáš, Black Jack, závody formulí...
- Zaměříme se na hrací část, ne na vstup a výstup.
- U her předpokládáme, že hrají rozumně se chovající jedinci (s motivací vyhrát).

Al-Capone a Babinský na sebe práskají

A,B	Babinský neudá	Babinský udá
Al-Capone neudá	0,0	25,0
Al-Capone udá	0,25	10,10

Optimum je udat!

Shannonova věta

Theorem (Shannon)

Každá kombinatorická hra má pro některého z hráčů neprohrávající strategii.

Důkaz.

Náznak: Buďto platí, že si jeden z hráčů může vynutit zacyklení hry (a tak neprohrát), nebo budeme zkoumat predikáty:
Existuje náš tah, že pro každý tah protihráče existuje náš tah, že pro každý tah protihráče... protihráč prohraje.
Pro každý náš tah existuje tah protihráče, že pro každý náš tah... my prohrajeme.
Formule jsou konečné, počty tahů jsou také konečné, jsou to vzájemně negace a lze je algoritmicky rozhodnout. □

Graf hry

- Ke hře (případně její instanci) definujeme orientovaný graf:
- Vrcholy: Stavy hry,
- Hrany: Možnosti přechodů mezi jednotlivými stavy.
- Příklad pro Nimm, kdy odebíráme 1 nebo 2 sirky (na tabuli).
- Každému stavu můžeme přiřadit barvu říkající, zda se odtud vyhrává nebo prohrává.

Příklad grafů her

- Na hracím plánu tvaru orientovaného grafu vyrážíme z určeného vrcholu. Taháme jedním padesátníkem.
- Máme dojet do jednoho z cílových vrcholů. Kdo dojede, vyhraje.
- Graf hry máme přímo zadaný a jde jen o to, který vrchol vyhrává.
- Podivná hra: Graf hry si zakreslíme na šachovnici. Vrcholy jsou políčka, hrany vedou tudy, kudy může figurka.
- Stačí říct, ze kterého vrcholu se vyhrává, prohrává, nebo zda existuje cyklus, po kterém mají oba hráči zájem bloudit (resp. zda si někdo z hráčů může bloudění po této kružnici vynutit).

AND-OR stromy

- Máme-li graf konečné hry, můžeme z něj postavit strom odpovídající hře.
- V tomto stromě nás zajímá, zda existuje větev, po které pokud pojedeme, tak vyhrajeme.
- Tato větev se pozná tak, že ve všech synech jejího koncového vrcholu existuje vyhrávající cesta, tedy ...
- buďto vyhrajeme v prvním synu, nebo ve druhém synu, nebo ve třetím...
- V k-tém synu vyhrajeme, jestliže protihráč prohraje v prvním synu a současně ve druhém synu a současně ve třetím synu... tohoto stavu.
- Prohraje tam, jestliže my (pro dotyčný stav) umíme vyhrát buďto v prvním synu, nebo ve druhém, anebo ve třetím...
- Podmínky AND a OR se stále střídají, proto AND-OR strom.

Hry s ohodnocením

Definition

Hra s ohodnocením je taková hra, kdy cílové stavy jsou ohodnoceny číslem. Jeden hráč se pokouší výsledek maximalizovat, druhý minimalizovat.

Definition

Hra s nulovým součtem je taková hra, ve které zisk jednoho hráče je roven ztrátě druhého hráče.

Algoritmus MINIMAX

- Algoritmus lze použít pro hry s ohodnocením.
- Postavíme strom hry.
- Začneme od koncových vrcholů.
- Hodnota podstromu je minimum resp. maximum z hodnot synů (podle toho, zda hraje minimalizující nebo maximalizující hráč).

Algoritmus NEGAMAX

- Varianta algoritmu MINIMAX pro hry s nulovým součtem:

$$\max_{i \in S} -f(i) = \min_{i \in S} f(i).$$

- Jde vlastně o totéž, je ovšem jednodušší na naprogramování.

Heuristiky

- Obvykle se pokoušíme neprohledávat zbytečně všechno, pokud najdeme jednu možnost výhry, nemusíme hledat i všechny ostatní.
- α-β-prořezávání: Umíme-li v nějakém synu S vyhrát aspoň α a najdeme v některém následujícím synu T, že protihráč nás umí dotlačit na méně, nemá smysl vrchol T dále zkoumat.
- Pro opačný případ se používá β: Pokud nás nepřítel umí zatlačit na nejvýš β a v jiném synu mu utečeme přes, nemá smysl ten druhý syn zkoumat dále.

Reálné hry

Šachy, dáma, halma, mlýn...

- Můžeme postavit strom hry, ten je ale příliš velký.
- Nasadíme proto všelijaké heuristiky. Ty dosavadní ale stejně daleko nevedou.
- Statická ohodnocovací funkce: Funkce, která se pokouší odhadnout, zda je pozice perspektivní (dobrá) nebo ne.
- Prohledáváme strom hry jen po nějakou dobu (do nějaké hloubky). Na nalezené (neterminální) pozice nasadíme statickou ohodnocovací funkci.
- U šachů například můžeme počítat materiální převahu a body za ohrožené figurky (Colossus na Atari kolem roku 1985).

Konec

...děkuji za pozornost...

Otázky?