SPA teorija iz 1. kolokvija

Luka Radiček

January 12, 2015

A1 Riješite rekurzivnu jednadžbu $a_n = a_{n+1} - a_{n-1}$ pri čemu su $a_0 = 1$ i $a_1 = 1$.

Radi se of **Fibonaccijevom** nizu čije je analitičko rješenje

$$F_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^k - \left(\frac{1-\sqrt{5}}{2} \right)^k \right]$$

gdje je $k = n$, ako je $F_0 = 0$. Naš niz počinje od F_1, pa je $k = n + 1$.

A2

stog ATP kod kojeg se ubacivanje i izbacivanje elemenata obavlja na jednom kraju koji se zove vrh (LIFO - 'zadnji unutra, prvi van'). Npr. hrpa tanjura.

char U C-u cjelobrojni tip podataka veličine 1B u koji spremamo kodove za ASCII set znakova. Općenito, najmanji tip u koji možemo smjestiti neki set znakova.

simbol koji nije slovo U ASCII tablici svi znakovi koji nisu velika slova 65-90 ili mala slova 97-122.
Je li simbol slovo u C-u možemo provjeriti funkcijom isalpha() iz <ctype.h> ili direktnom provjerom nalazi li se *char* u jednom od traženih intervala.

A3

rječnik ATP sličan ATP set, ali sadrži samo operacije na jednom skupu (make_null, insert, delete, member).

pokazivač na pokazivač Vrijednost mu je adresa nekog drugog pokazivača. (tako možemo unedogled)

```
int *p,
    **q,
    ***r,
    x = 10;

p = &x;
q = &p;
r = &q;

*** r == 10
** r == &p
* r == &q
```

Koristi se u implementaciji ATP rječnik pomoću otvorene rasute tablice. **Dictionary pokazuje na polje (zaglavlje) koje sadrži pokazivače na početke vezanih listi (pretince).

B1 Riješite rekurzivnu jednadžbu $a_n = na_{n-1}$ pri čemu je $a_0 = 1$.

$$
\begin{array}{ll}
a_0 = 1 & \\
a_1 = a_0 = 1 & \\
a_2 = 2a_1 = 2 & n \\
a_3 = 3a_2 = 6 & n\,(n-1) \\
a_4 = 4a_3 = 24 & n\,(n-1)\,(n-2) \\
a_5 = 5a_4 = 120 & n\,(n-1)\,(n-2)\,(n-3)
\end{array}
$$

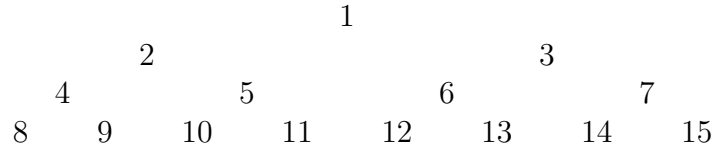
$$
a_n = \prod_{k=0}^{n-1} (n-k) \quad , \quad n > 0
$$

Analitičko rješenje je **n!**

rekurzivna jednadžba Jednadžba koja rekurzivno definira niz, jednom kada su zadani jedan ili više početnih članova. Svaki sljedeći član niza definiran je kao funkcija prethodnih.

rekurzivna funkcija Funkcija koja poziva samu sebe.

B3 Imate binarno stablo na kojem su oznake



Je li to puno ili potpuno stablo? Puno, a time ujedno i potpuno.

Kojim će se redosljedom ispisivati oznake ako stablo obilazimo koristeći funkcije preorder / inorder / postorder?
preorder: 1 2 4 8 9 5 10 11 3 6 12 3 7 14 15
inorder: 8 4 9 2 10 5 11 1 12 6 13 3 14 7 15
postorder: 8 9 4 10 11 5 2 12 13 6 14 15 7 3 1

Napišite funkciju koja obilazi stablo na način preorder i objasnite što se treba promijeniti da obilazak bude postorder, a što da bude inorder.

```
struct Node {
    char data;
    struct Node *left;
    struct Node *right;
};

void Preorder(struct Node *root) {
    if(root == NULL) return;
    printf("\%c ", root->data); // ispisuje
    Preorder(root->left); // prelazi na lijevo dijete
    Preorder(root->right); // prelazi na desno dijete
}
```

Mijenjamo samo redosljed radnji u funkciji.
Postorder se prvo spušta po lijevim podstablima do lista.
*left i *right od lista pokazuju na NULL, pa taj poziv funkcije ispisuje *data* od lista. Prethodni poziv ispisuje desni list (podstablo) ukoliko postoji itd.

```
    Postorder(root->left);
    Postorder(root->right);
    printf("\%c ", root->data);
```

```
    Inorder(root->left);
    printf("\%c ", root->data);
    Inorder(root->right);
```

binarno stablo Stablo koje se sastoji od nijednog, jednog ili više čvorova drugog stupnja. Svaki čvor može imati lijevo i desno podstablo.

potpuno stablo - sve razine osim zadnje popunjene su čvorovima
- čvorovi zadnju razinu popunjavaju s lijeva na desno
- čvorovi su numerirani redom s lijeva na desno od korijena prema zadnjoj razini

puno stablo Specijalan slučaj potpunog stabla kojemu je i zadnja razina popunjena čvorovima.

obilaženje stabla Algoritam kojim posjećujemo čvorove stabla tako da svaki čvor posjetimo točno jednom. Zadaje se rekurzivno.

C1

a priori vremenska složenost Brzina kojom raste vrijeme izvođenja algoritma kao funkcija relevantnih argumenata. Najjednostavnije ju je iskazati 'Big O' notacijom (najgori mogući slučaj). Nezavisna je od računala / programskog jezika / compilera. (teorija)

a posteriori složenost Brzina kojom raste vrijeme izvođenja algoritma dobivena mjerenjem na računalu i statistikom. (praksa)

algoritam za sortiranje Algoritam koji permutira neki niz podataka $(a_1, a_2, ..., a_n)$ čije elemente možemo uspoređivati totalnim uređajem $\leq$, tako da nakon permutiranja vrijedi $a_1 \leq a_2 \leq ... \leq a_n$.

C2 Riješite rekurzivnu jednadžbu $a_n = 4a_{n-1} - 3a_{n-2}$ pri čemu su $a_0 = 0$ i $a_1 = 1$.

$$\begin{array}{ll} a_0 = 0 & \\ a_1 = 1 & 3^0 \\ a_2 = 4 & 3^0 + 3^1 \\ a_3 = 13 & 3^0 + 3^1 + 3^2 \\ a_4 = 40 & 3^0 + 3^1 + 3^2 + 3^3 \end{array}$$

$$a_n = \sum_{k=0}^{n-1} 3^k \quad , \quad n > 0$$

Analitičko rješenje je **red potencija**.