Dynamické programování I cvičení

1.

- Popište, jak byste výpočet hodnoty rekurzivní funkce f(6,7) převedli na vyplňování hodnot v poli, když funkce f je rekurzivně definována takto:

a)

f(x,y) = 0 pro x=0 nebo y=0

f(x,y) = f(x–1, y–1) + f(x–1,y) + f(x,y–1) + 1 jinak

b)

f(x,y) = 0 pro x=0 nebo y=0

f(x,y) = max (f(x–1, y–1)+ f(x–1,y)) + f(x,y–1) + 1 jinak

2.

- Popište, jak převedete výpočet hodnoty rekurzivní funkce f(6,8,7) na vyplňování hodnot v poli:

a)

f(x,y,z) = 0 pro x=0 nebo y=0 nebo z=0

f(x,y,z) = f(x–1, y–1, z–1)+ f(x–1, y, z)

+ f(x, y–1, z) + f(x, y, z–1)+1 jinak

b)

f(x,y,z) = 0 pro x=0 nebo y=0 nebo z=0

f(x,y,z) = 3*f(x–1,y–1,z)–f(x,y–1,z)–f(x–1,y,z–1)+1 jinak

3.

- Pro kombinační čísla (= ***Bin***omické koeficienty) platí

  Bin(n,k) = ***Bin***(n-1,k)+***Bin***(n-1,k-1).

Funkce BIN(n, k) implementuje binomický koefiocient:

BIN(n,k) = 1 pro n = 0 nebo k = 0

BIN(n,k) = Bin(n–1,k)+Bin(n–1,k–1) jinak

Určete, kolikrát byla funkce BIN() volána při provedení příkazu

int x = BIN(6,4);.

4.

- Hodnoty Ackermannovy funkce A(n,m) lze standarně zapisovat do tabulky.

  A(n, m) = m+1 pro n=0

  A(n, m) = A(n–1, 1) pro n>0, m=0

  A(n, m) = A(n–1, A(n,m–1)) pro n>0, m>0

- Popište, jak budete postupně vyplňovat tabulku, chcete-li se vyhnout rekurzivnímu volání.
- Dokážete určit hodnotu A(4,4)?

5.

- Optimální binární vyhledávací strom

a) minimalizuje hloubku stromu,
b) maximalizuje cenu uzlů,
c) maximalizuje počet listů,
d) minimalizuje dobu vyhledávání ve stromu,
e) minimalizuje délku cesty z kořene do libovolného listu.

6.

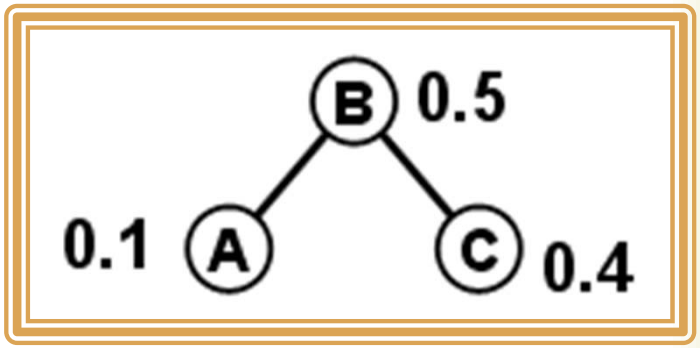
- Je dáno n klíčů. Společně s každým klíčem je dána také pravděpodobnost dotazu na tento klíč. Nalezení kořene optimálního BVS sestaveného z těchto klíčů má asymptotickou složitost

a) $O(\log(n))$
b) $\Theta(n)$
c) $O(n \cdot \log(n))$
d) $\Omega(n^2)$
e) $\Omega(2^n)$

7a.

- Pravděpodobnost dotazu na konkrétní klíče daného BVS je uvedena u jednotlivých uzlů. Dlouhodobě je průměrný počet navštívených uzlů při jednom dotazu na klíč ve stromu roven

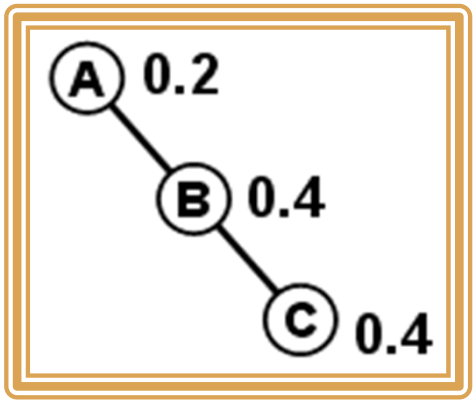
a) 0.5
b) 1.0
c) 1.25
d) 1.5
e) 1.75



7b.

- Pravděpodobnost dotazu na konkrétní klíče daného BVS je uvedena u jednotlivých uzlů. Dlouhodobě je průměrný počet navštívených uzlů při jednom dotazu na klíč ve stromu roven

a) 0.2
b) 1.0
c) 2.15
d) 2.2
e) 2.5

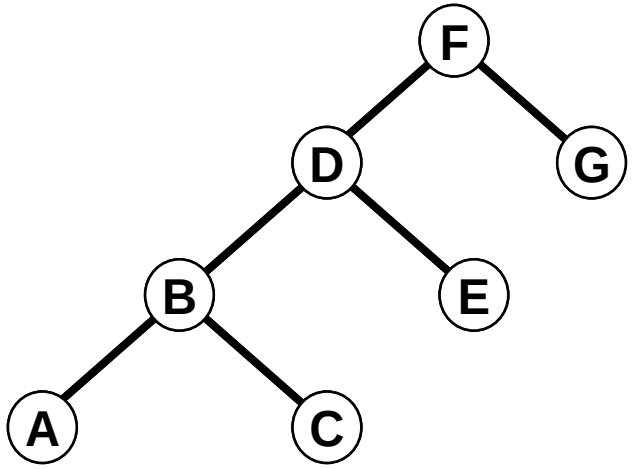


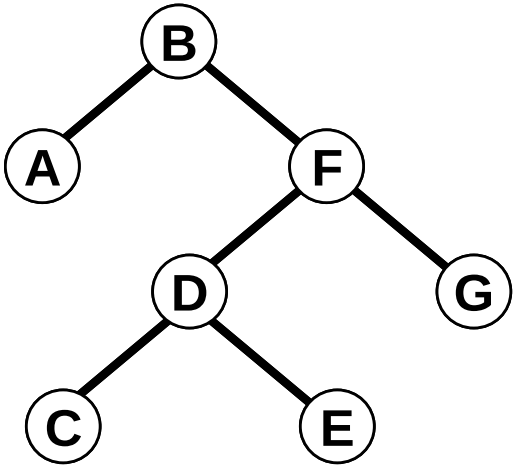
8.

- Pravděpodobnost dotazu na jednotlivé klíče v obou daných BVS je tato:

A: 0.10 B: 0.20 C: 0.25 D: 0.05 E: 0.10 F: 0.25 G: 0.05

- Vypočtěte, který strom je výhodnější pro operaci FIND.





9.

- Určete, jak bude vypadat optimální BVS, když jej vybudujeme pro 7 klíčů s frekvencemi:

a)

E 0.04 F 0.05 G 0.22 H 0.04 I 0.06 J 0.05 K 0.15

b)

A 0.10 B 0.10 C 0.25 D 0.35 E 0.10 F 0.05 G 0.05

10.

- Dva dané řetězce mají oba délku n. Nejdelší společnou podposloupnost těchto řetězců lze nalézt pomocí dynamického programování v čase

a) $\Theta(\log(n))$
b) $\Theta(n)$
c) $\Theta(n \cdot \log(n))$
d) $\Theta(n^2)$
e) $\Theta(n^3)$

11.

- Najděte nejdelší společnou podposloupnost dvojice řetězců

a) A: 1100110011001100
   B: 1010101010101010

b) A: 11010010001000010000 1000001
   B: 00101101110111101111 0111110 (B = doplněk A)

c) A: 11010010001000010000 1000001
   B: 10000010000010000100 0100101 (B = A pozpátku)

12.

- V dané matici začneme kdekoli v prvním sloupci a dále vždy postupujeme jen ve směru SV, V, JV až kamkoli do posledního sloupce. Cena cesty je součet hodnot navštívených polí. Jaká může být nejmenší?

29	10	11	23	22	23
27	25	29	12	29	24
18	21	11	27	14	24
30	17	26	29	23	22
12	25	23	13	28	16
20	24	10	14	30	15

13.

- Maticí cestujeme podle stejných pravidel jako v předchozí úloze. Cena za cestu je součet absolutních hodnot rozdílů cen sousedních navštívených polí.

29	10	11	23	22	23
27	25	29	12	29	24
18	21	11	27	14	24
30	17	26	29	23	22
12	25	23	13	28	16
20	24	10	14	30	15