Започнат на	Monday, 31 January 2022, 13:00
Състояние	Завършен
Приключен на	Monday, 31 January 2022, 13:59
Изминало време	58 мин. 20 сек.

Въпрос **1**

Отговорен

От максимално 2,00

Обяснете как се отразява Data Locality на операциите със свързан списък.

Обяснете какъв е ефектът върху основните операции и коментирайте дали той е положителен или негативен.

Свързаният списък не може да се възползва от data locality по същия начин, по който един масив би могъл, понеже указателите, които свързаният списък държи в себе си могат да сочат на произволно място в паметта, а не в една последователна част от тях, както при масива.
Тази особеност забавя операциите върху списъка, тъй като трябва да се "разхождаме" из паметта, за да намерим следващия елемент.
Списъкът държи в себе си както стойността на елемента си, така и указател към следващия елемент, което в някой случаи може да заема повече памет, отколкото биха заели същите данни, поставени в масив.
Плюсът на това, че не работим в един последователен блок от памет е, че ако искаме да добавим нов елемент на к-та позиция в списъка, не трябва да се съобразяваме с това дали имаме достатъчно място за този елемент в блока от памет, в който работим, както при един масив.

Въпрос **2**

Отговорен

От максимално 1,00

Нека е даден **сортиран** масив, който съдържа N-елемента.

Каква ще бъде сложността на всяка от дадените по-долу операции с масива?

Забележка: В списъка с отговорите със знак ^ е обозначена операцията степенуване. По-конкретно:

- O(N^2) *обозначава сложността* $O(N^2)$
- O(2^N) *обозначава сложността* $O(2^N)$

Проверка дали даден елемент се съдържа в масива (с изчерпващо търсене)	O(N)
Намиране на максималния елемент в масива	O(1)
Извличане на стойността на елемент намиращ се на даден индекс *к* в масива	O(1)
Добавяне на елемент на първа позиция в масива	O(N)
Проверка дали даден елемент се съдържа в масива (с двоично търсене)	O(log(N))
Вмъкване на елемент на произволна позиция	O(N)

Въпрос **3**

Отговорен

От максимално 1,00

Вярно ли е, че всяко двоично наредено дърво за търсене (binary search tree) е също и двоична пирамида (binary heap)?

Изберете едно:

- Истина
- Лъжа

Въпрос **4**

Отговорен

От максимално 1,00

Ако трябва да намерим най-дългия (като брой дъги) път между два върха в граф, можем ли да използваме за тази цел алгоритъма за обхождане в широчина (breadth-first search)?

Изберете едно:

- Истина
- Лъжа

Въпрос **5**

Отговорен

От максимално 1,00

Нека е даден **едносвързан** списък, който съдържа N-елемента. Считаме, че представянето е такова, че разполагаме с указатели към първата и последната кутия в списъка.

Каква ще бъде сложността на всяка от дадените по-долу операции със списъка?

Забележка: В отговорите със знак ^ е обозначена операцията степенуване. По-конкретно:

- O(N^2) *обозначава сложността* $O(N^2)$
- O(2^N) *обозначава сложността* $O(2^N)$

Вмъкване на елемент на първа позиция в списъка	O(1)
Намиране на максималния елемент в списъка	O(N)
Вмъкване на елемент на последна позиция в списъка	O(1)
Изтриване на първия елемент на списъка	O(1)
Вмъкване на елемент на произволна позиция в списъка	O(N)
Изтриване на последния елемент на списъка	O(1)
Проверка дали даден елемент се съдържа в списъка	O(N)

Въпрос **6**

Отговорен

От максимално 1,00

Нека имаме имплементация на separate chaining или linear probing хеш съдържащ N елемента. Ако трябва да намерим най-малкия или най-големия елемент в нея, това в общия случай е операция със сложност:

Изберете едно

- a. $O(2^N)$
- b. $O(1)$
- c. $O(\log N)$
- d. $O(N^2)$
- e. $O(N)$

Въпрос **7**

Отговорен

От максимално 2,00

Нека е даден Separate Chaining хеш, който съхранява цели числа и използва хешираща функция еквивалентна на следното:

```
unsigned int hash(int x) {
    return abs(x % 10);
}
```

Дайте пример за такава поредица от числа, която би предизвикала серия от колизии и в резултат всяко добавяне (или търсене) на число от поредицата ще бъде със сложност O(N), вместо с O(1). Тук N е броят на числата съхранени в хеша.

Ако разгледаме поредица от числа, чиито модул по 10 е винаги 2 или (-2), то всички ще се хешират с една и съща стойност и търсенето би било със сложност O(N). Същото би се случило и ако голяма част от поредицата от числа има еднакъв абсолютен модул по 10, а друга не. За другата част търсенето ще остане със сложност O(1) или малко по-голяма.

Въпрос **8**

Отговорен

От максимално 1,00

Нека изпълняваме бързо сортиране (quicksort) върху масив с N-елемента.

Делителния елемент (pivot) за всеки pass избираме да бъде първият елемент в частта от масива, която се сортира.

Кое/кои от следните условия водят до най-лошия случай на изпълнение - $O(N^2)$?

Изберете едно или повече:

- a. Всички (или почти всички) елементи в масива са еднакви
- b. Масивът е (почти) сортиран в обратен ред
- c. Масивът е (почти) сортиран
- d. Избран е делителен елемент (pivot), който е прекалено голям (например най-големият елемент в масива).
- e. Избран е делителен елемент (pivot), който е прекалено малък (например най-малкият елемент в масива).

Въпрос **9**

Отговорен

От максимално 1,00

Нека е дадено произволно двоично наредено дърво за търсене (binary search tree), което съдържа N-елемента.

Каква е сложността на търсенето, вмъкването и премахването на елемент в най-лошия случай?

Изберете едно

a. $O(1)$

b. $O(\log N)$

c. $O(N)$

d. $O(N \log N)$

e. $O(N^2)$

f. $O(2^N)$

Въпрос **10**

Отговорен

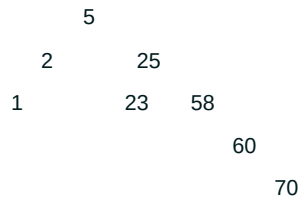
От максимално 2,00

Обяснете накратко какво означава за едно двоично наредено дърво да бъде "изродено".

Обяснете как израждането влияе върху операциите с дървото.

Ако едно двоично дърво е изродено, това значи, че едната му част ще съдържа много малък брой елементи, а всички останали ще попаднат в другата - която ще се изроди в даден момент в едносвързан списък. Основните операции върху дървото са със сложност O(logN), когато то е балансирано. При наличието на изродено дърво (едносвързан списък някъде в дървото) тези операции вече ще са със сложност O(N), което нарушава хубавите свойства на добре нареденото двоично дърво.

Пример:



Въпрос **11**

Отговорен

От максимално 1,00

В общия случай кои от следните са предимства на балансираните двоични наредени дървета пред хешовете?

Изберете едно или повече:

- a. Ако искаме да обходим съхранените елементите в нарастващ ред, това става по-бързо в дървото.
- b. Добавянето на елемент става по-бързо в дървото.
- c. Всички хешове имат максимален брой елементи N, които могат да се запишат в тях, докато дървото може да нараства колкото поискаме.
- d. Ако искаме да намерим всички елементи от интервала [A, B], които се съдържат в структурата, това става по-бързо в дървото.
- e. Ако имаме стойност N и искаме да намерим най-близкия по стойност елемент съхранен в структурата, това става по-бързо в дървото.
- f. Търсенето на елемент става по-бързо в дървото.

Въпрос **12**

Отговорен

От максимално 1,00

Вярно ли е, че алгоритъмът за обхождане в дълбочина (depth-first search) винаги има нужда от повече памет за работата си, отколкото алгоритъма за обхождане в широчина (breadth-first search)?

Изберете едно:

- Истина
- Лъжа

Въпрос **13**

Отговорен

От максимално 1,00

Нека е дадено празно двоично наредено дърво за търсене *T*.

В дървото добавяме числата от 1 до 9 в следния ред: 7, 1, 3, 5, 4, 9, 2, 6, 8.

Ако обхождаме дървото в ред ляво-корен-дясно (in-order) и извеждаме числата на екрана, какво ще се изведе на екрана?

(От падащите менюта изберете стойностите така, че да се получат числата точно в реда, в който ще се изведат на екрана)

4	5	6	2	3	8	7	1	9

Въпрос **14**

Отговорен

От максимално 1,00

Трябва да създадем контейнер, за който имаме следните изисквания:

1. Да можем да добавяме произволен брой елементи към контейнера;
2. Добавянето да бъде със сложност $O(1)$ (може да бъде амортизирана);
3. Няма нужда елементите да се поддържат в някакъв определен ред

Кои от следните структури от данни може да се използват за реализирането му?

Изберете едно или повече:

- [x] a. Свързан списък с една връзка
- [ ] b. Динамичен масив
- [ ] c. Двоично наредено дърво
- [ ] d. Статичен масив
- [x] e. Хеш
- [ ] f. Пирамида (heap)
- [x] g. Свързан списък с две връзки

Въпрос **15**

Отговорен

От максимално 1,00

Вярно ли е, че дадено по-долу дърво е двоично дърво за търсене (binary search tree)?

```
1
 \
  2
   \
    3
     \
      4
```

Изберете едно:

- (•) Истина
- ( ) Лъжа

Въпрос **16**

Отговорен

От максимално 1,00

Кое от следните е необходимо, за да може да приложим алгоритъма за двоично търсене (binary search) върху даден масив?

Изберете едно

- ( ) a. Масивът да има не повече от 2^32 елемента
- (•) b. Масивът да е сортиран
- ( ) c. Никое от посочените -- алгоритъмът може да се приложи върху произволен масив
- ( ) d. Масивът да не съдържа отрицателни числа
- ( ) e. В масива да няма повтарящи се елементи
- ( ) f. Масивът да съдържа цели числа

Въпрос **17**

Отговорен

От максимално 1,00

Трябва да създадем контейнер, за който имаме следните изисквания (N е броят на елементите, които се съдържат в контейнера):

1. Намирането на k-ия най-малък елемента да става със сложност O(1);
2. Намирането на k-ия най-голям елемента да става със сложност O(1);
3. Търсенето за произволен елемент да става със сложност не по-голяма от O(log N);

Коя от следните структури от данни може да се използва за реализирането му?

Изберете едно

- a. Сортиран списък с две връзки
- b. Динамичен масив
- c. Двоично наредено дърво
- d. Сортиран динамичен масив
- e. Пирамида (heap)

Въпрос **18**

Отговорен

От максимално 3,00

Обяснете накратко каква е разликата между свързан списък и двоично дърво за търсене. След това посочете какви са предимствата на дървото пред списъка и на списъка пред дървото (ако има такива).

Свързаният списък не ни дава никакви математически връзки между елементите си - той ги пази в последователността, в която ги добавим към него.

Двоичното дърво за търсене разглеждата стойносттите на елементите, които биват добавени в него. Ако стойността, която добавяме е по-малка от стойността в корена, то тя бива добавенав вляво от него, а ако е по-голяма - отдясно. Това правило се прилага рекурсивно при изграждането на дървото. Благодарение на него ние имаме наредба на елементите.

Предимства на списъка пред двоичното дърво:

- При добавяне на елементи в края или в началото на свързания списък сложността е O(1), докато при дървото добавянето на елемент е със сложност O(logN), понеже се търси мястото на елемента в дървото;
- При изтриване на елементи в края или в началото на свързания списък сложността е O(1). изтриване на произволен елемент от дървото е със сложност O(logN) в общия случай;

Предимства на дървото пред свързания списък:

- При добавяне на елемент на произволна позиция сложността при дървото е O(logN) , а при свързания списък O(N);
- Същото важи и за изтриване на елемент на произволна позиция;
- Търсенето на елемент в дървото отново е със сложност O(logN), а в списъка O(N);
- Дървото се "сортира" още при създаването си, докато списъкът трябва да бъде допълнително сортиран, ако не се създаде сортиран.

Въпрос **19**

Отговорен

От максимално 1,00

Нека са дадени четирите числа 1, 5, 10 и 100. По колко начина тези числа могат да бъдат наредени в двоично дърво, така че то да бъде двоично дърво за търсене (binary search tree)?

Отговор: 24

Въпрос **20**

Отговорен

От максимално 1,00

По-долу е даден кодът на функцията Mystery, която сортира масив от цели числа. Кой алгоритъм използва функцията?

```
void Mystery(int* pArr, size_t Size)
{
    size_t i = Size / 2;

    while (i--)
        Help(pArr, i, Size);

    i = Size;

    while (--i)
    {
        std::swap(pArr[0], pArr[i]);
        Help(pArr, 0, i);
    }
}

void Help(int* pArr, size_t pos, size_t Size)
{
    int elem(pArr[pos]);

    size_t ni = pos;
    size_t si = pos * 2 + 1;

    while (si < Size)
    {
        if (si < Size - 1 && pArr[si] < pArr[si + 1])
            si++;

        if (elem > pArr[si])
            break;

        pArr[ni] = pArr[si];
        ni = si;
        si = si * 2 + 1;
    }

    pArr[ni] = elem;
}
```

Изберете едно

a. Броене на честоти (Counting sort)

b. Пирамидално сортиране (Heap sort)

c. Сортировка на Шел (Shell sort)

d. Сортиране чрез сливане (Merge sort)

e. Функцията не реализира никой от алгоритмите посочени в другите отговори

f. Метод на мехурчето (Bubble sort)

g. Сортиране чрез клатене (Shaker sort)

◄ Шаблон за практическата част

Отиди на ...

Списък с препоръчвани източници за C/C++ ►