Алгоритми за сортиране

Лекция 2 по СДА, Софтуерно Инженерство
Зимен семестър 2018-2019г
Милен Чечев

План на днешната лекция

- Организационни въпроси.
- Алгоритми за сортиране
- Контролно 1

Резултати от входяща анкета

Времето на провеждане на лекциите удобно ли ви е и ще можете ли да ги посещавате?

Отговор	Средно	Общо
Да	87%	140
Не	13%	21
Общо	100%	161/161

Работите ли?

Отговор	Средно	Общо
Да	27%	44
Не	73%	117
Общо	100%	161/161

Разполагате ли с лаптоп, който да може да носите за лекциите по структури от данни и алгоритми?

Отговор	Средно	Общо
Да	93%	149
Не	7%	12
Общо	100%	161/161

Подготвени ли сте за курса?
За да отговорите положително трябва като минимум да знаете какво е масив, как да използвате операторите for и if, как да получавате и записвате данни от конзола или файл, как да дефинирате клас/структура.

Отговор	Средно	Общо
Да	95%	153
Не	5%	8
Общо	100%	161/161

Известен ли ви е сайта https://www.hackerrank.com/ и решавали ли сте задачи в него?

Отговор	Средно	Общо
Да	35%	57
Не	65%	104
Общо	100%	161/161

Занимавали ли сте се със състезателно програмиране?

Отговор	Средно	Общо
Да	11%	18
Не	89%	143
Общо	100%	161/161

Мислите ли, че представеният курс ще ви е полезен за професионалната ви реализация?

Отговор	Средно	Общо
Да	99%	159
Не	1%	2
Общо	100%	161/161

Ще можете ли да отделяте в допълнение към 3-те часа лекции и 2-та часа упражнения още 2-3 часа на седмица за самостоятелна подготовка вкъщи?

Отговор	Средно	Общо
Да	93%	149
Не	7%	12
Общо	100%	161/161

Интересувате ли се от летен стаж в голяма технологична компания(Гугъл, Фейсбук, Амазон,...)?

Отговор	Средно	Общо
Да	76%	122
Не	24%	39
Общо	100%	161/161

Знаете ли английски език на ниво достатъчно за да четете техническа литература за програмиране на него?

Отговор	Средно	Общо
Да	92%	148
Не	8%	13
Общо	100%	161/161

Самостоятелното писане на програмен код е от изключителна важност за развитие на уменията за програмиране и решаване на алгоритмични проблеми.

С участието си в този курс се съгласявате да не участвате в схеми за преписване или измама. С подобни действия в дългосрочен план ще ощетите хората, на които евентуално краткосрочно "помагате". Ако искате да помогнете на някой обяснявайте му извън лекции, как се решават задачите, но не му ги решавайте. Потвърждавате ли своето съгласие?

Отговор	Средно	Общо
Да	100%	161
Не		0
Общо	100%	161/161

Обратна връзка

25 попълнени форми - благодаря на всички за отделеното време.

Избрани коментари:

- “оценяването може да се оптимизира - за студентите от горен курс е изключително натоварващо”,
- “може ли да се уточни какво ще е самото точкуване за домашните работи, и по-конкретно за брой преминати тестове(точки, проценти) за достигане на удовлетворяващ резулат преди изпита.”
- “домашните изискват много повече време от 2 часа.”
- “Малко хумор няма да навреди на никого.”

Резултати от първо състезание за домашно

https://www.hackerrank.com/contests/practice-1-si/leaderboard

Следващи стъпки

- Езици за програмиране, на които може да се пишат контролни и домашни: C++, Java, C#
- Отбелязване на задачите за домашно като задължителни, препоръчителни и допълнителни.
- Допитване за откриване на допълнителни групи за упражнения за студентите от минали години

Първо контролно

- Инструкции:
  - Заредете предварително лаптопите си поради ограничената електрическа мощност в залата.
  - Проверете свързаността ви с интернет. (По време на контролното ще има около 200 включени компютъра и е много вероятно Wi-Fi мрежата да не издържи, използвайте мобилен интернет ако имате такава възможност)
  - Пригответе се да седнете спрямо инструкциите(два пълни реда 1 празен?)
  - По време на контролното не е позволено ползването на програми за комуникация или търсене в гугъл.
  - Не се опитвайте да преписвате или да помагате на някой (Така проваляте възможността да се научи!)
  - Ако сте приключили с контролното - излезте от стаята без да вдигате шум

Алгоритми за сортиране

Какво ще научим?

1. Какво е сортиране и за какво ни е необходимо?
2. “Бавни” алгоритми за сортиране (bubble, selection, insertion)
3. “Бързи” алгоритми за сортиране (merge, quick)
4. Специфични алгоритми за сортиране(count sort)

Какво е сортиране и за какво ни е необходимо?

- Основен клас алгоритми много често необходим за решаване на реални проблеми
- Добър пример за демонстриране на това какво е алгоритъм, как се изчислява сложност на алгоритъм, как да бъдем по-критични към това с каква сложност решаваме проблем.

Дефиниция на сортиране

Проблем: Да се напише процедура, която подрежда в нарастващ ред обекти подадени като вход.

Вход: масив с числа

Изход: масив със същите числа наредени в нарастващ ред.

Сортиране с метод на мехурчето

Основна идея:
Започвайки последователно от началният елемент до крайният сравняваме всеки елемент със следващия като ги разменяме ако не са подредени. По този начин на всяка стъпка изкарваме най-големият в края на правилното за него място. Повтаряйки този процедура толкова пъти колкото е големината на масива постигаме правилно подреждане на целият масив.

Сортиране с метод на мехурчето

```
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}
```

Сортиране с пряка селекция

Намаля броя на разменянията в сравнение с метода на мехурчето!

Основна идея: Търсим най-големият елемент в масива и директно го поставяме на последно място. След това следващият по големина и отново го поставяме на място и т.н. Докато всички се подредят.

Сортиране с пряка селекция

```
for (j = 0; j < n-1; j++){
	int iMin = j;
	for (i = j+1; i < n; i++){
		if (a[i] < a[iMin]){
			iMin = i;
		}
	}

	if (iMin != j){
		swap(a[j], a[iMin]);
	}
}
```

Сортиране с вмъкване

Основна идея: сортиране постепенно на все по-голяма част от масива, като обхождайки несортираната част всеки един елемент го поставяме в сортираната част с намиране на правилното за него място на което сортираният масив остава сортиран.

Сортиране с вмъкване

```
void insertionSort(int arr[]) {
   for (int i = 1; i < arr.length; i++) {
       int key = arr[i];
       j = i-1;
       while (int j >= 0 && arr[j] > key) {
           arr[j+1] = arr[j];
           j = j-1;
       }
       arr[j+1] = key;
   }
}
```

Demo

https://visualgo.net/en/sorting

Сложност на сортиращите алгоритми

Бавни сортировки:

Bubble, Selection, Insertion - O(n^2)

Бързи сортировки:

Merge, Quick - O(n*log(n))

Сортиране чрез сливане(merge sort)

Основна идея: Ако имаме два сортирани масива то със линейна сложност може да ги влеем в един масив. Тогава ако разделим масива който искаме да сортираме на по-малки масиви и на всяка стъпка сливаме два по-малки масива в един голям то за log(N) стъпки ще слеем всички масиви до един масив, като всяка от стъпките е била линейна.

Сортиране със сливане реализация

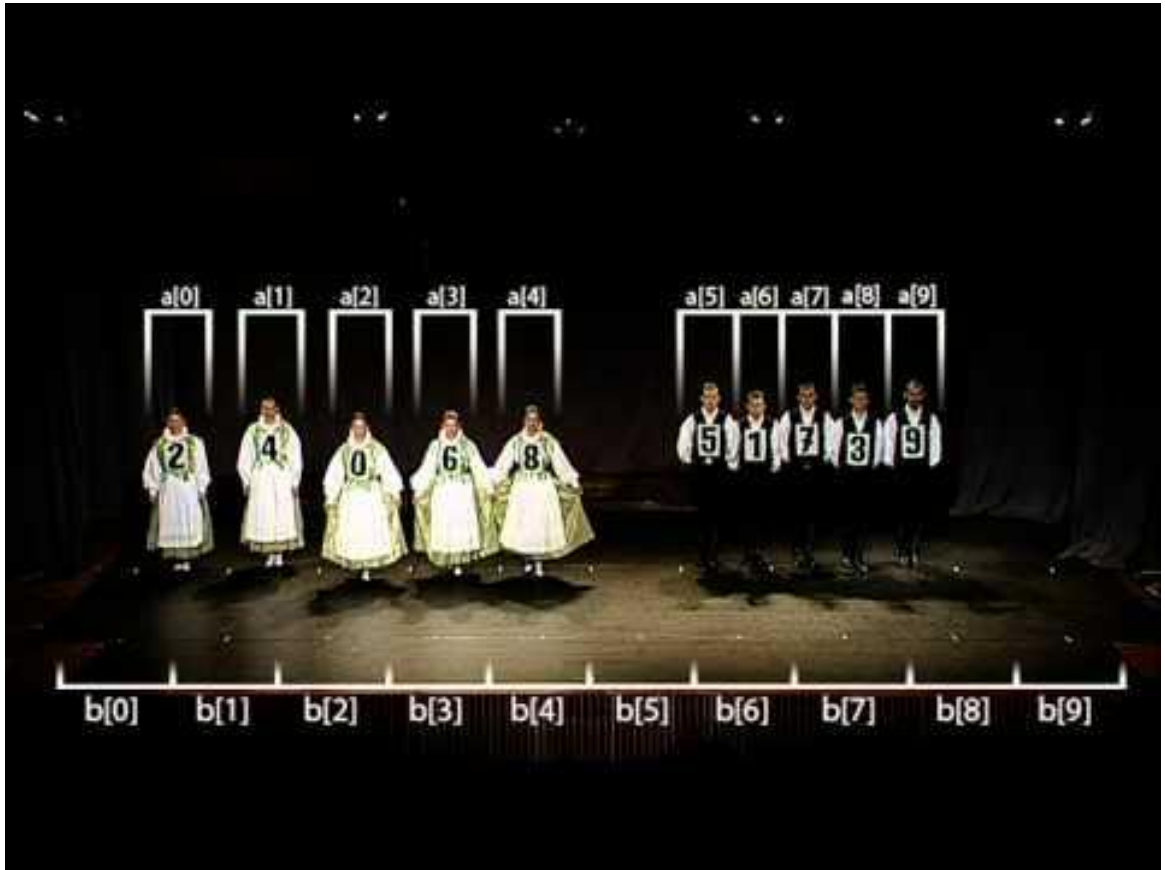
```
void mergesort(int arr[], int l, int r)
    {
        if (l < r) //гранично условие на рекурсията
        {
            mergesort(arr, l, (l+r)/2);
            mergesort(arr , (l+r)/2+1, r);
            merge(arr, l, m, r); //функция която слива два масива
        }
    }
```

Сортиране със сливане реализация(2)

```
void merge(int arr[], int start, int middle, int end){
    // Създаваме масивите arr1 и arr2, които съдържат частите, които ще копитаме. По този начин си освобождаваме основният масив за презаписване.

    i = 0, j = 0, k = start;
    while( i < arr1.length; j < arr2.length){
        // по малкото число от arr[i] и arr2[j] го записваме в  arr[k]
        // увеличаваме брояча на масива от който копирахме
        K++; // увеличаваме брояча за основният масив
    }
    // допълваме с всички останали необходени елементи от arr1 и arr2
}
```

Сортиране със сливане(Забавно видео)



Бързо сортиране (quick sort)

Основна идея: Ако вземем едно произволно число от масива, то с линейна сложност можем да прехвърлим всички по-малки числа от масива да са в ляво на числото, а всички по-големи в дясно.

При бързото сортиране избираме число от масива прехвърляме по-малите отляво, по-големите от дясно и след това изпълняваме същата процедура за лявата и дясната половина.

По тази процедура получаваме сложност в средният случай O(n*log(n))

Бързо сортиране реализация

```
void sort(int arr[], int low, int high)
    {
        if (low < high)
        {
            int pi = partition(arr, low, high);

            sort(arr, low, pi-1);
            sort(arr, pi+1, high);
        }
    }
```

Бързо сортиране реализация(2)

```
int partition(int arr[], int low, int high){
        int pivot = arr[high];
        int i = low; // index of smaller element
        for (int j=low; j<high; j++){
            if (arr[j] <= pivot){
                swap(arr, i, j)
                i++;
            }
        }

        swap(arr, i, high)
        return i;
    }
```

Сортиране с броене (Counting sort)

Можем ли да сортираме със сложност по-малка от O(n*log(n))?

Отговор: Да, но с добавяне на допълнителни ограничения.

При сортирането с броене сложността се определя от броя на различните елементи, които може да има в масива.

Сортиране с броене

Основна идея: Понеже имаме ограничен брой различни стойности в масива то може да преброим по колко пъти се среща всяка една от тези стойности(с едно обхождане на масива) и след това със второ обхождане да наредим стойностите по техният ред.

Стабилност на сортирането - ако имаме два елемента които са равни в първоначалният масив, то във финалният те се срещат във същият ред като първоначалният масив.

```
void conting_sort(char arr[]) {
        char arr_copy[] = new char[arr.length];
        for (int i = 0; i<arr.length; ++i) {
            arr_copy[i] = arr[i];
        }
        int count[] = new int[256];
        for (int i=0; i<n; ++i) {
            count[arr[i]] = count[arr[i]]+1;
        }
        for (int i=1; i<=255; ++i) {
            count[i] += count[i-1];
        }
        // To make it stable we are operating in reverse order.
        for (int i = n-1; i>=0; i--) {
            arr[count[arr_copy[i]]-1] = [arr_copy[i];
            count[arr_copy[i]] = count[arr_copy[i]] - 1;
        }
}
```

Сложност на сортиране с броене

O(n+k)

Време за контролно

Обърнете внимание на инструкциите на следващите два слайда!

Важно!

Контролните и задачите за самостоятелна работа са изключително важен процес за развиване на уменията за решаване на алгоритмични проблеми, не се допуска никакво преписване! Ако някой има нужда от помощ, може да му помогнете с обяснение как да си реши задачата(извън залата за контролно), но не и да му давате наготово код за решението на контролно или домашно.

Контролно 1

- Започва в 17:15, време за работа 45 мин.
- Инструкции:
  - Заредете предварително лаптопите си поради ограничената електрическа свързаност в залата.
  - Проверете свързаността ви с интернет. (По време на контролното ще има 200 включени компютъра и е много вероятно Wi-Fi мрежата да не издържи, използвайте мобилен интернет ако имате такава възможност)
  - Пригответе се да седнете спрямо инструкциите(два пълни реда 1 празен?) Първите редове са за хората които ще пишат на хартия като те се нареждат през едно място.
  - По време на контролното не е позволено ползването на програми за комуникация или търсене в гугъл.
  - Не се опитвайте да преписвате или да помагате на някой (Така проваляте възможността да се научи!)
  - Ако сте приключили с контролното - излезте от стаята без да вдигате шум