Теми за държавен изпит

18. Структури от данни. Стек, опашка, списък, дърво. Основни операции върху тях. Реализация.

Структури от данни:

СД са схеми за организиране на даден вид данни в паметта, обикновено с цел ефикасност. Всеки съставен тип данни (ТД) в частност може да се разглежда като СД.

пример: int[3 е масив, но може да се разглежда и като СД. В същност масивите са едни от най-простите СД.

Абстрактните типове данни (АТД) са формален модел на ТД или СД, които дават логическото им описание (предназначение, операции, компоненти). Не налагат конкретна организация на паметта и конкретно представяне в даденя език.

Видове СД:

- според вида на елементите
  - хомогенни (масив)
  - хетерогенни (структура)
- според способността за промяна на размера
  - статични (статичен масив)
  - динамични (вектор)
- според връзките между елементите
  - линейни (списък)
  - ~~разклонени (дърво)~~ йерархични (дърво)
  - мрежови (граф)

Сложност на алгоритъм:

Алгоритъм е по-ефикасен, ако има нужда от по-малко ресурс. Разглеждаме два вида сложност:

- времева – оценка на време за изпълнение
- пространствена – оценка на използвана памет.

Оценката, която разглеждаме, е асимптотична. Т.е. интересуваме се какво се случва при големи данни ($n \to \infty$). Ще ползваме O-нотацията.

Стек:

АТД стек е хомогенна линейна структура от данни с организация "последен влязъл – пръв излязъл" (LIFO). Основните операции са:

- create() – създаване на празен стек
- empty() – проверка дали е празен
- push(x) – добавяне на елемент
- pop() – премахване на елемент
- peek() – преглед на последния влязъл елемент

Характеристики и реализация: на купа

Опашка:

АТД Опашка е хомогенна линейна структура с организация "пръв влязъл – пръв излязъл" (FIFO). Основните операции са:

- create()
- empty()
- enqueue(x)
- dequeue()
- head()

Списък:

АТД Списък е хомогенна линейна структура с последователен достъп до елементите. Основните операции са:

- create()
- empty()
- get(p) – достъп до елемент на позиция p.
- insert(x, p) – вкл. на елемент x на позиция p
- delete(p) – изкл. на елемент на позиция p

Дървета:

АТД дърво е йерархична структура, в която на всеки елемент е съпоставено множество от подчинени елементи.

Кореново дърво е списък $(X, T_1, \dots, T_n)$, където

- X е корен (данна)
- $T_1, \dots, T_n$ са кореновите поддървета, $n \geq 0$

Основни операции:

- create(x) – създаване на дърво с корен x
- addSubtree(t) – добавяне на поддърво t
- root – достъп до корена
- subtrees() – достъп до поддърветата

Дефиниции за дърва:

- дете – корено възел на поддърво
- родител – възел, чиито деца е коренът
- листо – възел без деца
- път – редица от възли, в която всеки следващ е дете на предходния

Схеми на обхождане:

- префиксно – първо посещаваме корена, после децата
- постфиксно – първо обхождме децата, после корена

дървета с фиксиран брой деца – всеки възел има един и същи брой деца

Основни операции:

- empty_tree()
- create_tree(x, t_1, ..., t_n)
- root()
- subtree(i)

двоично дърво:

- $\perp$ (празно дърво)
- наредена тройка (X, L, R) – X е корен, L е ляво поддърво, R е дясно

Схеми на обхождане:

- префиксно $(K \Lambda \Phi)$
- инфиксно $(\Lambda K \Phi)$
- постфиксно $(\Lambda \Phi K)$

Двоично дърво за търсене:

Организация, която позволява бързо намиране на елементи в дървата. Разчита се на линейна наредба над елементите.

Основни операции:

- create()
- insert(x)
- remove(x)
- search(x)

дефиниция:

- $\perp$ е ДДТ
- (X, L, R) е ДДТ, ако $X >$ всеки елемент в L, $X <=$ всеки елемент в R, L и R също са ДДТ