Изпитна тема № 8: Алгоритми и структури от данни

**План-тезис:** Динамично оптимиране. Дървовидни структури от данни и алгоритми върху тях.

Хеширане и хеш-таблици. Графи и алгоритми върху графи.

**Динамично оптимиране:**

Метод за решаване на задачи с припокриващи се подзадачи.

Работи на принципа “Разделяй и владей”.

Изграждат се връзки, свързващи решението на задачата с решенията на по-малки подзадачи от същия тип.

Всяка една от по-малките подзадачи се решава само веднъж, резултата им се записва и се получава на решение на първоначалната задача.

**Дървовидни структури от данни и алгоритми върху тях:**

Представляват разклонени йерархични структури от данни, които са изградени от възли. Всеки възел е свързан с други възли (разклонения на дървото).

Дървовидни структури от данни са:

* Дървета - двоични, балансирани, подредени и др.
* Графи - ориентирани, неориентирани, с тегла и др.
* Мрежи - графи с особени свойства

Важно за дърветата:

* Всеки възел(или връх) може да има родител и наследник
* Възел без родител се нарича корен (root), всяко дърво има **само един корен**.
* Възел без наследници се нарича листо (leaf)
* Могат да се разделят на поддървета

Двоични дървета:

* Възлите имат не повече от **две разклонения**
* Няма правила за подредба, освен ако не са сортирани двоични дървета
* “Двоични дървета за търсене” - сортирани, лявото разклонение на възела има по-малка стойност от възела, а дясното има по-голяма. Могат да бъдат **балансирани дървета.**
* Балансирани дървета -
  + - всеки възел има  
      почти еднакъв брой възли във своите  
      поддървета
    - имат височина, приблизително равна на log(n)
* B-trees - Имат много наследници, добри за бази от данни.
* Червено-черни -
  + - Всички листа са черни и са null (нямат инфо в тях)
    - Корена е черен
    - Няма възел, който да има две червени връзки към него
    - Всеки път от даден възел до листо в негово поддърво има еднакъв брой черни възли
    - Червените възли са винаги от ляво

Алгоритми върху дървета:

Алгоритми за обхождане са по дълбочина (Depth First Search) или по ширина (Breadth First Search).

По дълбочина (DFS):

* Първо се посещават наследниците на възела.
* Прави се с рекурсия - първо се обработват децата на възела, ако няма такава се взима стойността на възела.

По ширина (BFS):

* Първо се посещава най-близкият възел, после неговите деца
* Прави се с опашка - добавя се възел, когато се махне от опашката, се записва стойността и се добавят неговите деца

**Хеширане и хеш-таблици:**

Хеширащите функции конвертират ключ от произволен тип до стойност от целочислен тип (например Иван - 259, Пешо - 361).

Хеш таблицата е стандартен масив, който съдържа набор от наредени двойки {ключ, стойност}

**Хеширане** - функцията, с която се определя кой ключ на коя позиция в масива да се съхрани

Перфектно хеширащата функция свързва всеки ключ към уникално цяло число в рамките на конкретен интервал

В повечето случаи перфектното хеширане е невъзможно

Свойства на добрата хешираща функция

* Консистентност - един и същ ключ трябва да произвежда един и същ хеш
* Ефективност - ефективни при изчисляването на хеш
* Равномерност - хешовете, произведени от хеширащата функция трябва да са равномерно разпределени

При генериране на еднакви ключове се случва колизия. Колизията се разрешава чрез:

* Свързване на елементите на колизията
* Използване на друга клетка от таблицата
  + Линейно пробване - взима се следващата празна клетка
  + Квадратично пробване - hash + 12, h + 22, h + 32,...
  + Двойно хеширане - повторно се използва hash функция

**Графи и алгоритми върху графи:**

Граф е абстрактна структура от данни и представлява връзките между отделните елементи на дадено множество. Всеки елемент от това множество се нарича **връх**, а връзката между два върха се нарича **ребро**.

Видове графи:

* **Ориентиран граф** - ребрата са насочени, изобразяват се чрез стрелки.
* **Ориентиран мултиграф** - ребрата са насочени. Възможно повече от едно ребро да свързва два върха. В него може да съществуват:
  + изолиран връх – в който не влизат и не излизат ребра
  + примка – ребро, чието начало и край съвпадат
  + повече от едно ориентирано ребро между два върха
* **Неориентиран граф** - ребрата нямат посока
* **Претеглен граф** - ребрата имат тегла

При графите, всеки връх съдържа списък със своите съседи и теглото до тях.

Алгоритмите за обхождане на графи са:

* DFS
* BFS
* На Дейкстра (Dijkstra) - За всеки връх се пази по едно число, показващо дължината най-късия път, намерен до момента, от старта до този връх. Отначало тези числа са равни на безкрайност, а в течение на работата на алгоритъма намаляват при откриване на все по-къси пътища.
* На Прим (Prim) - Избира се случаен връх и се маркира. След това се намира се най-близкият до него, немаркиран връх и също се маркира. Така, докато не се маркират всички върхове. Избраните пътища между тях са най-късите, край на алгоритъма.
* На Крускал (Kruskal) - Търси се минимално покриващо дърво в претеглен, свързан граф, като ацикличен подграф с ребра, на който сумата е минимална. Избират се трите най-малки ребра. След това се свързват, чрез най-малкото ребро между тях. Така, докато всички точки не са свързани в един подграф.

Условие на приложната задача: <https://drive.google.com/file/d/1fBrqUx2NckP-SjjInIwWsYV4r1H-xut8/view>

Практическо решение на приложната задача:

<https://github.com/viksuper555/Softuni-Finals-prep>