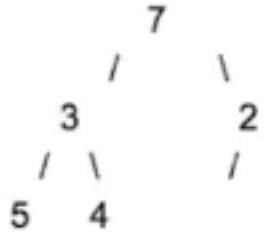
Class 5 Heap & Graph Search Algorithms I

堆（英语：**heap**）亦被称为：**优先队列**（英语：**priority queue**）

Example



insert ()

index 0	1	2	3	4	5
X	3	2	5	4	7

Heap: is an unsorted array but have special rules to follow

性质: 堆的实现通过构造二叉堆（binary heap）,这种数据结构具有以下性质

1. 任意节点小于它的所有后裔，最小元素在堆的根上（**堆序性**）。
2. 堆总是一棵完全树。complete tree
3. 将根节点最大的堆叫做MAX HEAP，根节点最小的堆叫做最小堆MIN HEAP
4. index of lChild = index of parent X 2 + 1
5. index of rChild = index of parent X 2 + 2
6. unsorted but follow rules above

堆（英语：binary-**heap**) 亦被称为：**优先队列**（英语：**priority queue**）

Example

index 0	1	2	3	4	5
1	3	2	5	4	7 0

Heap: is an unsorted array but have special rules to follow

性质: 堆的实现通过构造二叉堆（binary heap）,这种数据结构具有以下性质

1. 任意节点小于它的所有后裔，最小元素在堆的根上（**堆序性**）。
2. 堆总是一棵完全树。complete tree
3. 将根节点最大的堆叫做MAX HEAP，根节点最小的堆叫做最小堆MIN HEAP
4. index of lChild = index of parent X 2 + 1
5. index of rChild = index of parent X 2 + 2
6. unsorted but follow rules above

支持的基本操作

1. insert：向堆中插入一个新元素； **时间复杂度O(log(n))**
2. update：将新元素提升使其符合堆的性质； **时间复杂度O(log(n))**
3. get/top：获取当前堆顶元素的值； **时间复杂度O(1)**
4. pop：删除堆顶元素； **时间复杂度O(log(n))**

支持的基本操作

1. insert：向堆中插入一个新元素； **时间复杂度O(log(n))**
2. update：将新元素提升使其符合堆的性质； **时间复杂度O(log(n))**
3. get/top：获取当前堆顶元素的值； **时间复杂度O(1)**
4. pop：删除堆顶元素； **时间复杂度O(log(n))**

5. **heapify**：使得一个unosrted array变成一个堆。 **时间复杂度O(n)**
   5.1. https://en.wikipedia.org/wiki/Heapsort

https://en.wikipedia.org/wiki/Heapsort – Change | Remove

Method 1: sort it! and return first k elements. O(n^2)

Method 2 : min-heap

Step1: heapify the whole array to make it a MIN-Heap. O(n)

Step2: keep popping k times → O(k log(n))

Method 3 : max-heap

Step1: **insert first k elements into the max-heap**

Step2: for k+1-th element to the n-th element, and for each new element X

if X < max-heap.top(). we call max-heap.pop() and call max-heap.insert(X)

else do nothing

O(k + (n-k)log(k))

Method 2 : min-heap

Step1: heapify the whole array to make it a MIN-Heap. O(c*n)

Step2: keep popping k times → O(k log(n))

Method 3 : max-heap

Step1: **insert first k elements into the max-heap**

Step2: for k+1-th element to the n-th element, and for each new element X

if X < max-heap.top(). we call max-heap.pop() and call max-heap.insert(X)

else do nothing

O(k + (n-k)log(k))

	M2		M3
	O(n + (k log(n)))		**O(k + (n-k)log(k))**
Case1 n>>>>k	**O(c* n)**	is hard to say	**O(nlog(k))**
Case2 n ~ k	O(nlogn)	is hard to say	O(nlogn)

Method 4 : quick-partition

Borrow the idea from quickSort k = 7 n =1000

xxxxxxxxxxxxxxxxx P1 pivot = p1

xxxxxxxxP2 xxxxxxxxx pivot = p2

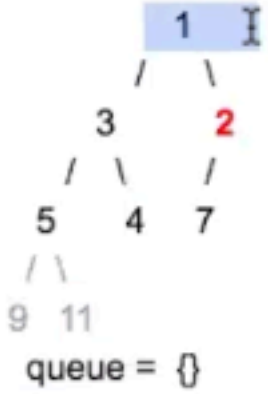
XXP3 xxxxxxxxxxxxxx pivot = p2

XXP3 xxxxxxxxxxxxxxx k = 7-3 = 4

n + n/2 + n/4 + … ⇒ **O(2n) = O(n)** average time complexity. However, in the worst case, O(n^2)

图里常用的search 算法

1. **Breadth-First Search (BFS-1)** :

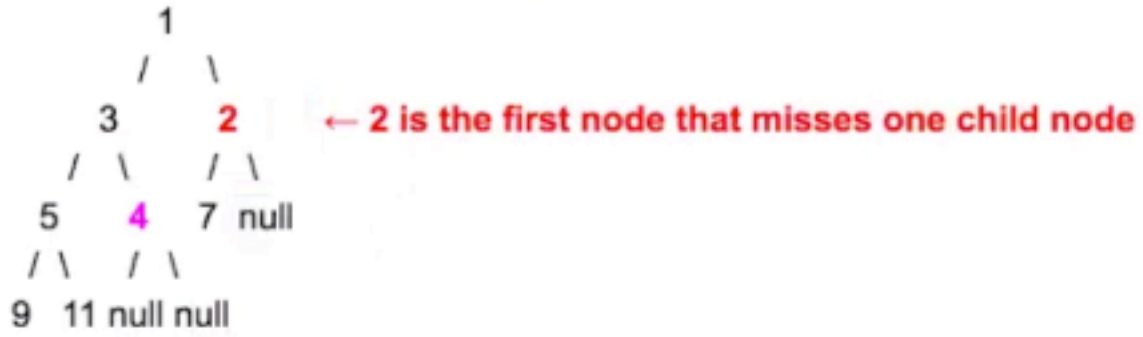


expand(1) → generate(3) and generate(2)

BFS的操作过程 & How to describe a BFS's action during an interview?

- **Definition 1**: **expand** a node s: **中文：延展一个node， e.g.** visit/print its value….
- **Definition 2: generate** s's neighbor node: reach out to its neighboring node (Fisst, to generate Node 3, and then generate Node 2).
- **Data Structure**: Maintain a **FIFO queue**, put all generated nodes in the queue. e.g., 3 and then 2 into the queue (FIFO) queue head-> [3, 2] tail
- **Termination condition**: do a loop until the queue is empty
- **Process**:

经典例题3： Determine whether a binary tree is a complete binary tree



Case1: if we found a node that misses its left child (right child != null) return false;

Case2: after detecting the first node that misses one child, then check whether all following nodes expanded to see whether they have any node generated (if any → then false)

DISCUSSION:

1. **When to consider using BFS1?**
    a. **When we are solving the relationship among the nodes in the same level**
2. **Is BFS1 the right solution to find shortest path in a graph??**
    a. **No! (answer yes if the graph has uniform edge cost)**

2. Best First Search (BFS-2)

经典算法：Dijkstra's Algorithm (runtime efficiency improvement: A* algorithm https://en.wikipedia.org/wiki/A*_search_algorithm unnecessary to read)

1. **Usages**: Find the shortest path cost from a single node (source node) to any other nodes in that graph (**点到面(==所有点)的最短距离算法**)
2. **Example problem**: 从北京到中国其他所有主要城市的最短距离是多少
3. **Data structure**: priority_queue (MIN_HEAP)
4. **解题思路**
   - 4.1. Initial state (start node)
   - 4.2. Node expansion/Generation rule:
   - 4.3. Termination condition: 所有点都计算完毕才停止,也就是 p_queue 变空\
5. **Example**

- 5.1. start node is **4**
- 5.2. cost(node) = cost(parent of node) + c(parent of node, node)

2.1. one node can be expanded once and only once

2.2. one node can be generated more than once. (cost can be reduced over time)

2.3. all the cost of the nodes that are expanded are monotonically non-decreasing (所有从priority queue里面pop出来的元素的值是单调非递减 --> 单调递增)

2.4. time complexity, for a graph with n node and the connectivity of the node is http://en.wikipedia.org/wiki/Dijkstra's_algorithm O(nlogn)

2.5. when a node is popped out for expansion, its value is fixed which is equal to the shortest distance from the start node.

3. Q1 **variant**, how to find the shortest path between a pair of node.

   3.1. **termination condition**: when the target node is expanded.

4. Q2 use its properties

   4.1.1. time complexity, for a graph with n node and the connectivity of the node is http://en.wikipedia.org/wiki/Dijkstra's_algorithm

   4.2. Q1 **variant**, how to find the shortest path between a pair of node.

      4.2.1. termination condition: when the target node is expanded.

经典考题：(运用 Dijkstra's Algorithm的性质)

Given a matrix of size NxN, and for each row the elements are sorted in an ascending order.
and for each column the elements are also sorted in an ascending order.
How to find the k-th smallest element in it?

e.g.,
[0][0]

```
12345
23456
34567
45678
56789
```

Solution:

解题思路

1. Initial state (start node) input[0][0]
2. Node expansion/Generation rule: Expand **[i][j]**
   generate [i][j+1]
   generate[i+1][j]
3. Termination condition: 所有点都计算完毕才停止,也就是 p_queue 变空\

经典考题：(运用 Dijkstra's Algorithm的性质)

Given a matrix of size NxN, and for each row the elements are sorted in an ascending order.
and for each column the elements are also sorted in an ascending order.
How to find the k-th smallest element in it?

e.g.,
[0][0]

```
12345
23456
34567
45678
56789
```

Solution:

解题思路

1. Initial state (start node) input[0][0]
2. Node expansion/Generation rule: Expand **[i][j]**
   generate [i][j+1]
   generate[i+1][j]
3. Termination condition: 所有点都计算完毕才停止,也就是 p_queue 变空\
   When the k-th element is popped out of the p-queue, then it must be the k-th smallest element.

经典考题：(运用 Dijkstra's Algorithm的性质)

Given a matrix of size NxN, and for each row the elements are sorted in an ascending order.
and for each column the elements are also sorted in an ascending order.
How to find the **k-th smallest** element in it?

e.g.,

[0][0]

```
123c45
23a456
3b4567
45678
56789
```

Time = O(klog(k))

Because, for each iteration, we need to pop 1 node out of thq p-queue O(logk)

in the meantime, we need to generate 2 nodes and insert them into the p-queque

O(2logk)

Thus, for each iteration, the time 3 logk

There are totally k iterations, and therefore the total time = O(k logk)