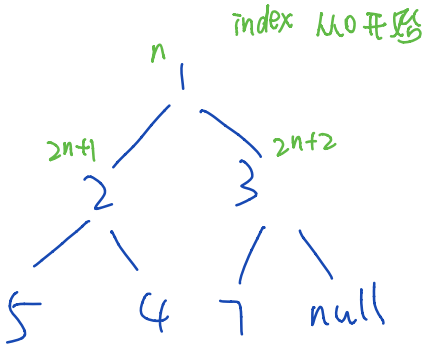
Heap 堆 (min/max). = PriorityQueue 优先队列

维护一个变化数据集的最优值

Complete BT: 气泡只能出现在最后一行的右侧.



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

左子树找 parent: $\frac{n-1}{2}$

右子树找 parent: $\frac{n-1}{2}$

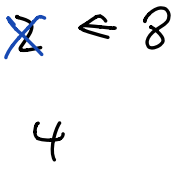
1. Heap is always a complete binary tree.

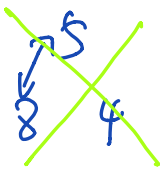
2. Every node is smalled than all its descendent.
   [Root 最小] ← For MinHeap
   后裔. (堆序性)

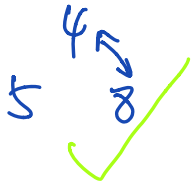
Insert: 放在气泡位置, 再和 root 依次交换.

Time: O(logn)

Update:







MinHeap 要和小的话.

Time: O(logn) 定点打击，告诉 index，并更新的情况下.

删掉顶端元素. pop

思考

1、如果一个一个往下删，容易破坏 tree 的结构. 不complete了.

2、把最后一个元素放到 root 上，再一个一个往下换

逻辑上是 Tree，物理上是 array.

可以有重复元素

一般不 search.

Q1. Find smallest k elements from unsorted array of size n.

Solution 0: sort → O(n log n)

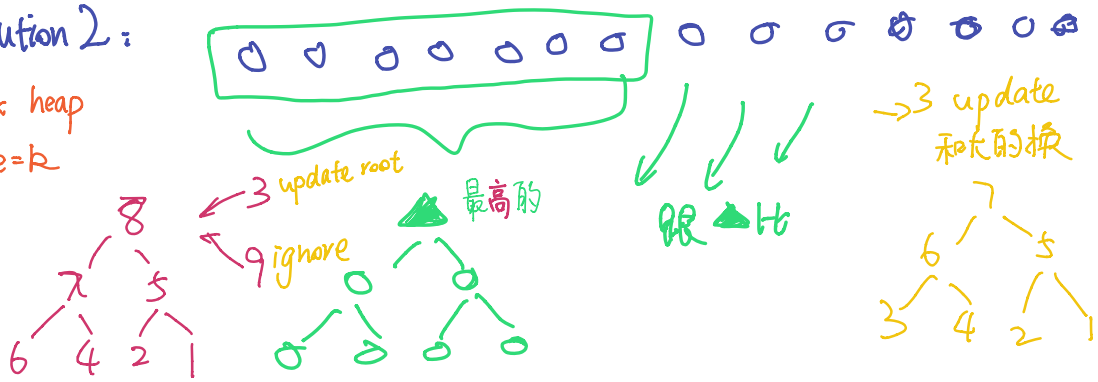
Solution 1:
min heap
size=n

step1: unsorted array —HeapSort, O(n)→ sorted array

step2: call pop() k times O(k log n)

⇒ Total O(n + k log n)

Solution 2:
max heap
size=k



step1: Heapify the first k elements O(k)

step2: Iterate

Case 1: >= root: ignore

Case 2: < root: update (root → new element)

step3: Finally, all elements remaining in the heap is the result.

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

Compare $O(n+k\log n)$ 和 $O(k+(n-k)\log k)$

$k \ll n$	$O(n * \text{constant})$	$O(n\log k)$	hard to say
$k \sim n$	$O(n\log n)$	$O(n+n\log n) = O(n\log n)$	hard to say

指任一个数量级.

hard

Space $O(n)$ > $O(k)$ ✓ better.

$O(n)$: Offline 要求得到所有 data 才能做

$O(k)$: Online 有 data streaming 时, better.

Solution): Quick Select: similar to quicksort.

$n = 1000$ xxxxxxx xxxxx P xxx xxxxxxx.

$k = 20$

Average time: $O(n + n/2 + n/4 + \cdots + \quad) = O(n)$

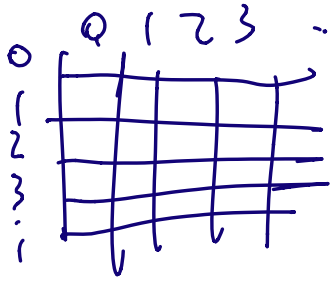
Worst time: $O(n + n-1 + n-2 + \cdots + \quad) = O(n^2)$

Graph 有向 directed

无向 undirected

表示方法

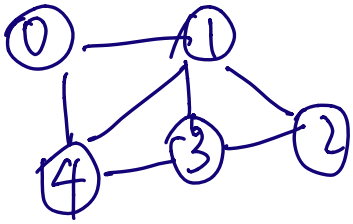
1. Adjacency Matrix



时间查找O(1)

浪费空间，尤其大部分情况为 sparse

2. Adjacency List



0 → 1 → 4

1 → 0 → 4 → 3 → 2

2 → ; 指上所有邻居

3 → |

4 →

缺点：查找不方便

需要 traverse the linked list

vertices

点：O(V)

边：O(E)

Edges

Space = O(V+E)

常见的 Graph Search Algo

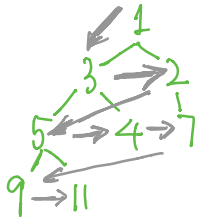
怎么 traverse 一张图.

1. Breadth-First Search (BFS-1)

宽度优先搜索算法

一层一层地毯式搜索

经典问题：分层打印一个 binary tree.



过程：

① 选一个 FIFO queue,

Queue = ← {2} ←

Queue {1}	expand (1) → generate (3) generate (2)
Queue {3, 2}	expand (3) → generate (5) generate (5) expand (2) → generate (7)
Queue {5, 4, 7}	expand (5) → generate (9) generate (11)

② Expand a node,

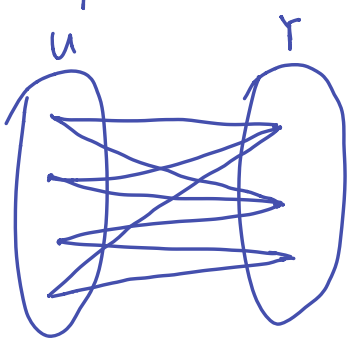
③ generate s's neighbor nodes,

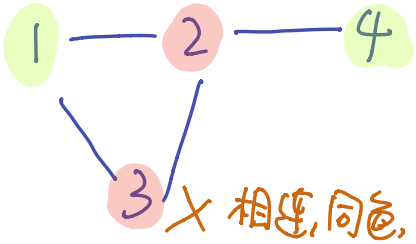
④ Termination Condition:

do a loop until the queue is empty.

换行问题？要先看要 expand 几个元素.

2. Bipartite 二分图





X 相连，同色，return false

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

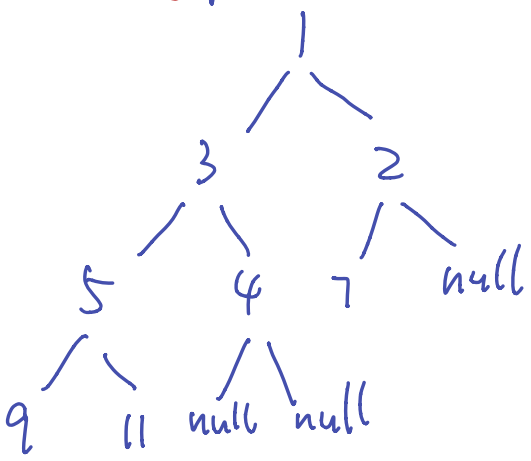
Queue = {2, 3}

expand (2) → generate (1) (3) (4)

如果全都上完色，没出现问题，return true

3. Complete binary tree BS判断.

定义：中间不能有气泡



expand	generate
1	3 2
3	5 4
2	7 null

一旦有气泡出现，后面不能再 generate 出任何数字了

```
if (generate null) {
    flag = true;
}
```

讨论：能用 BFS-1 解决最短路径问题吗？

Nope. 因为每条边加了权重，有 weight 了.

只有 every edge has the same weight，才可以用 BFS-1 找到最短路径.

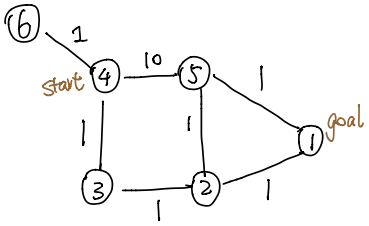
Best First Search (BFS-2) 找最短路径.

经典算法 Dijkstra's Algo. [假设 cost > 0]

Usage: Find the shortest path cost from a single node (source node) to any other nodes in the graph. 点到面

例子：从北京到中国其他所有主要城市的最短距离是多少. priority_queue (min-heap)

1. initial state
2. while() expansion/generation rule
   a. Expand a node
   b. Generate all neighbors
3. Termination condition



PQ = (3, 1)
(5, 10)
(6, 1)
expand (,)
expand (2, 2)
expand (1, 3)
expand (5, 3)

已经出现过了，找小的. 或者先比较，只把小的放进 heap

注意，即便 generate 出 goal 了，还要继续，因为可能有更小的.

性质：

1. one node can be expanded only once.
2. one node can be generated more than once.
3. 所有从 PQ 里 pop 出来的元素的值是单调递增的
4. Time: O(n log n) Dijkstra's e.g. 4-connected grid 曼哈顿棋盘格
   General graph, O((V+E) log V) = O(E log V)
5. 一旦一个 node 被 expand 了，它的最短路径就不会再变了.

经典考题 N×N. 每行排好序了. 找第k小的元素.

1	2	3	5	6
2	3	4	6	8
3	4	(5)	7	9
4	5	6	8	10
5	6	9	10	11

(5): i,j; 7: i+1,j; 6: i,j+1

Time = O(k log k)

Space = O(k + n^2) N×N

boolean array 要存下来

BFS2

1. initial state: array[0][0]
2. Expansion/Generation Rule:

   expand node[i][j]

   generate[i+1][j]

   generate[i][j+1]

3. Termination condition:

   when the k-th element is expanded

   (popped out of the PQ)

4. Deduplication: 重复的只generate一次.

   boolean[][] generated = true/false.