堆和优先队列

主讲：肖臻

优先队列（堆）

- 教材的相关章节
  - Algorithm Design，Chap2.5，英文版p57
  - Intro to Algo，2nd edition，Chap6.5，英文版p138（整个Chap 6都是讲heapsort）
- 维护堆结构的两个基本操作
  - 向上交换、向下交换

堆

- 为什么要“堆”这样的数据结构？
  - 所有操作用平衡二叉树都能完成
- 堆的数组实现
  - 完全二叉树的层次遍历
  - 堆中半数以上的节点是叶节点
  - 数组下标从0开始或从1开始，父节点和两个子节点的计算方法略有不同
  - 注意bit shifting并不改变变量本身的值
    - (i >> 1)+1不改变i的值

支持的操作

- 删除堆顶元素
  - 注意：如果允许删除堆中的任意元素，用堆中最后一个元素来替代，那么用来填空的元素既可能太大，也可能太小
  - 应用例子：子序列和最大，序列长度受限制的版本
- 动态改变优先级
  - 《算法导论》上最小堆只支持decrease_key，而最大堆只支持increase_key
  - Algorithm Design上面不区分，允许优先级两个方向改变

STL中的priority queue

- 实际应用中，堆中的元素都是struct复合变量
  - 需要重载<操作符
- 为什么不支持decrease key？
  - 怎样workaround？这样做对复杂度有什么影响？
  - 注意：STL中的关联容器也不允许修改key值
- 不支持reserve操作（STL中的容器适配器都不支持）
  - 对很多应用来说，堆中最大元素个数是已知的
  - 编译错：priority_queue<int> pq; pq.reserve(n);
  - 没有很好的解决方法，除非hack底层的vector实现
  - 可以要求底层用deque而非vector来实现
  - 或者自己用vector来实现一个堆

自己实现支持修改key值的堆

- 维护两个映射
  - key -> event ID：简单（堆中元素包含这两个域）
  - event ID -> 堆中的对应位置：主要难点在这里
- 用什么样的数据结构来记录堆中元素的位置？
  - 如果event ID分布的空间是紧密排列的，用数组
    - 比如Prim算法，图中的顶点个数
  - 否则需要用散列表
- 只有三个地方会更新堆中元素的位置：
  - heapify up/down时交换堆中两个元素的位置
  - extra-min时把最后一个元素拷贝到第一个元素。原来第一个元素在堆中的位置变成invalid

用multiset实现类似的功能

- 注意不能用set：堆中可以有key值相同的元素
- 同样不支持reserve操作
  - 关联容器不是连续存储的，没有这个需求
- 时间复杂度
  - 查询最小值可在常数时间内完成
    - multiset<int> s; cout << *s.begin();
  - 删除最小元素的均摊代价是常数的
  - 改变key值仍然是最麻烦的
    - 无法通过event ID查询
    - 当存在key值相同的很多event时，效率比较低

图论中的优先队列

- MST、Dijkstra算法
  - 如果一开始就把所有顶点一起建堆，那么之后不会用插入操作
    - 如果自己实现堆的话，建堆时只要把源点做为数组第一个元素，其它节点简单放入数组中即可。
    - 支持decrease_key仍然需要用到heapify_up
  - 也可以当每个顶点变得可达时（距离值不是无穷大）再加入
  - 注意：取决于图中边的稀疏程度，堆并不总是效率最高的数据结构
    - 必须用邻接表来表示图中的边，才能发挥出堆的优越性

优先队列的其它应用

- 堆排序
  - 一般用最大堆：从堆顶取出最大元素后可以直接放到数组末尾的最终位置上
  - 初始建堆可在O(n)内完成，而不用一个个插入的O(nlgn)复杂度
- Event driven simulator
- Dynamic median
  - 不需要decrease key操作
  - 能不能就选取平衡二叉树中的根节点？