Dijkstra runtime revisited

Monday, November 21, 2022 1:57 PM

```
toCheckQueue = V (prioritized on routeDist)
cameFrom = empty map

for v in V:
  v.routeDist = inf
source.routeDist = 0

while toCheckQueue is not empty:                         → |V| times
  checkingV = toCheckQueue.removeMin()                   → O(log|V|)
  for neighbor in checkingV's neighbors:                 → |E|/|V| times
    if checkingV.routeDist + cost(checkingV, neighbor) < neighbor.routeDist:     O(1)
      neighbor.routeDist = checkingV.routeDist + cost(checkingV, neighbor)
      cameFrom.add(neighbor -> checkingV)
  X   toCheckQueue.decreaseValue(neighbor)   O(log|V|) ← in ideal world

backtrack from dest to source through cameFrom
```

O((|V|+E|)log|V|)

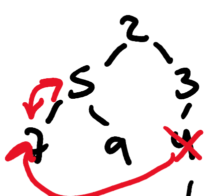
what we actually had to do in Project 2:

```
toCheckQueue.remove(neighbor)   ← arbitrary elt = O(|V|)
toCheckQueue.insert(neighbor)   O(log|V|)
```
] O(|V|)

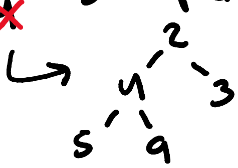
final Dijkstra runtime for project 2: O(|V|log|V| + |E|·|V|) → slower than ideal!

decrease 7 to 1:

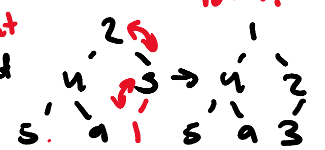
remove(7):



- search through heap to find 7 (linear)
- swap rightmost leaf in (constant)
- sift up or down as appropriate (log)

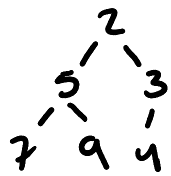


insert(1):
- insert 1 into first empty leaf (constant)
- sift up (log)

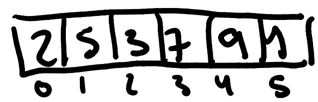


- if we have constant time access to the location of an elt, can decrease its priority in log N time. (while preserving the heap)

heap:



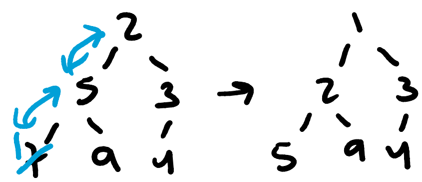
array representation:



map from elts. to indices (maintained w/ every insert/delete/sift operation):

5→1 4→5 9→4 7→3 2→0 3→2

decrease 7:
- find 7 (now constant!)
- decrease (constant)
- sift up (log n)



Key takeaways:
- Dijkstra runtime has potential of being O((|V| + |E|)log|V|)
- Java PQ doesn't have a "decrease" operation, so we have to find an element and insert it back in with a new priority, which is a linear operation (so Dijkstra ends up being O(|V|log|V| + |E||V|)
- Optimizing PQ with heaps that keep track of where each element is allows us to implement a logN decrease operation and get the better Dijkstra runtime