Priority Search Queue

2012.12.11

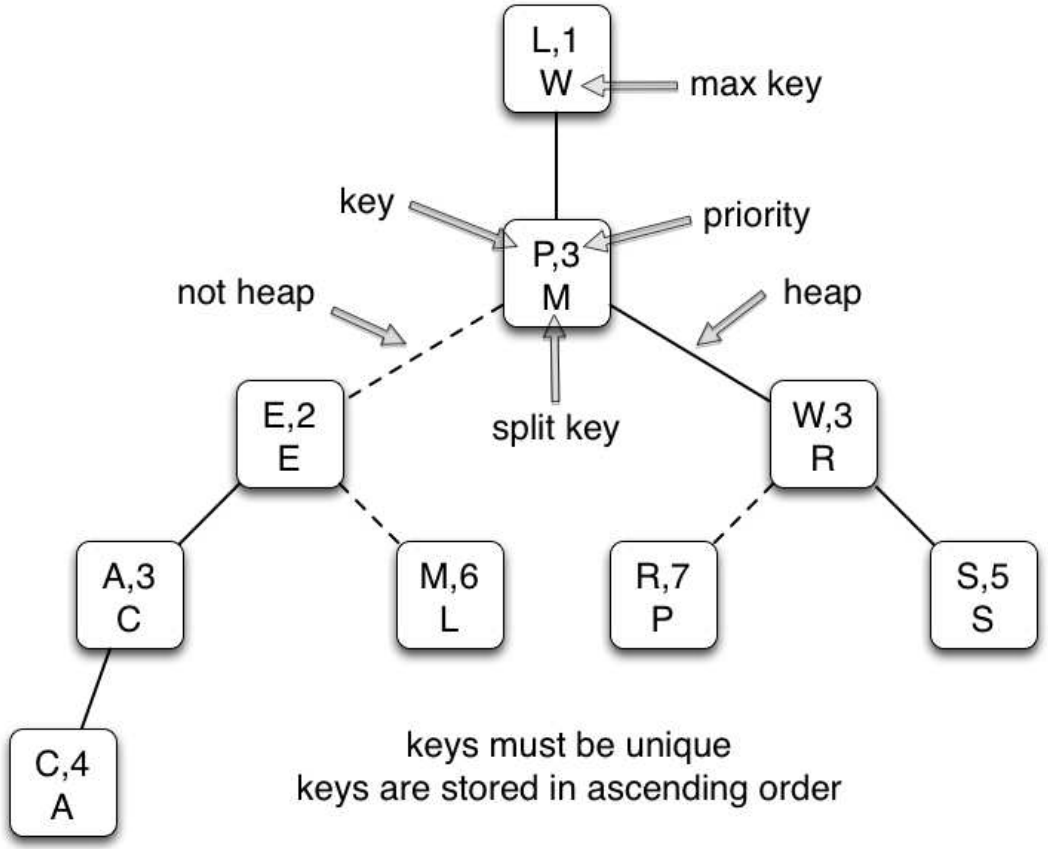
Kazu Yamamoto

What is PSQ?

- PSQ = Priority Search Queue
  - PSQ k p
- Search = Binary Search Tree with *key*
  - Priority as value (key - priority pair)
  - insert
    - k -> p -> PSQ k p -> PSQ k p
    - O(log n) in the worst case
  - lookup
    - k -> PSQ k p -> Maybe p
    - O(log n) in the worst case
- Priority Queue = Heap with *priority*
  - extractMin
    - PSQ k p -> Maybe (k, p, PSQ k p)
    - O(log n) in the worst case
  - atMost
    - p -> PSQ k p -> [Binding k p]
    - O(m log n) in the worst case where m is # of deleted items
  - updatePriority
    - (p -> p) -> k -> PSQ k p -> PSQ k p
    - O(log n) in the worst case

Example

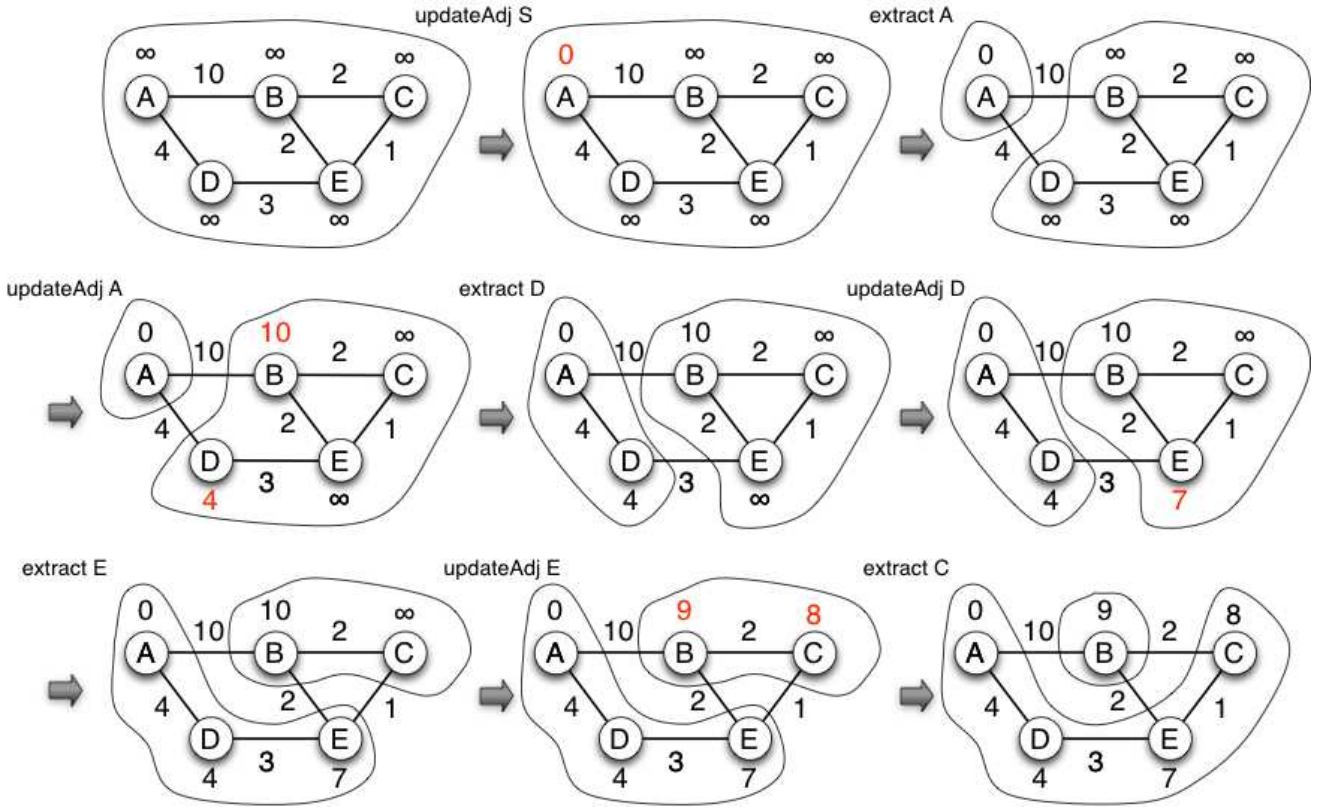
- A variant of loser tree or pennant



Applications of PSQ

- Timeout manager of Glasgow Haskell Compiler
  - Set timeout
    - insert with a unique key and a time as priority
  - Operation finished before timeout
    - delete with a unique key
  - Regular timeout
    - atMost to obtain expired entries
- Dijkstra's algorithm
  - Aka SPF (Shortest Path First)
  - Single source shortest path problem

Example of Dijkstra's algorithm



Dijkstra's algorithm

- Pseudo code

```
S ← ϕ
Q ← Vertces(G) with priority of ∞
updatePriority startPoint 0
while (Q ≠ ϕ) {
    u ← extractMin(Q)
    S ← S ∪ u
    foreach v ∈ Adj(u) {
        updatePriority v weight(u,v)
    }
}
```

- Cost

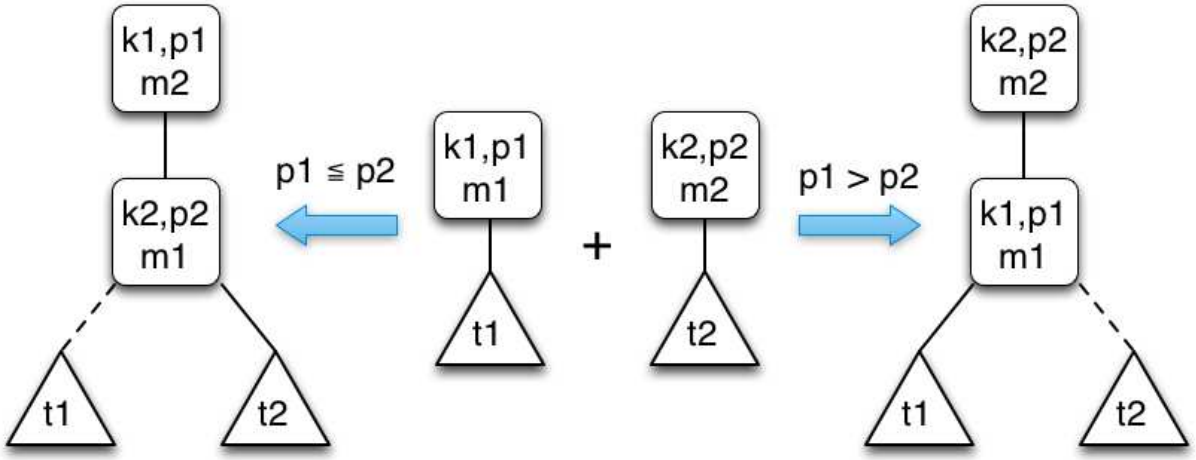
  |Vertex| * extractMin + |Edge| * updatePriority

Comparing other implementations

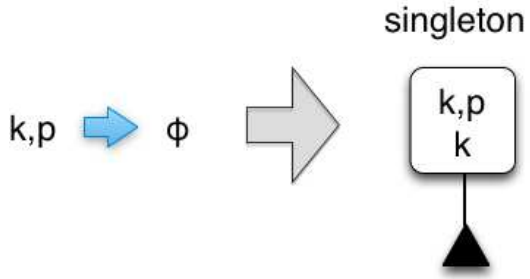
- Cost (again)
  - V * O of extractMin + E * O of updatePriority
- Array
  - $O(V^2 + E) = O(V^2)$
  - extractMin: O(V)
  - updatePriority: O(1)
- Binary heap with two arrays
  - One array to index with keys
  - O((V + E) log V)
  - extractMin: O(log V)
  - updatePriority: O(log V)
- PSQ
  - Purely functional (non-destructive)
  - O((V + E) log V)
  - extractMin: O(log V)
  - updatePriority: O(log V)

Construction and Destruction

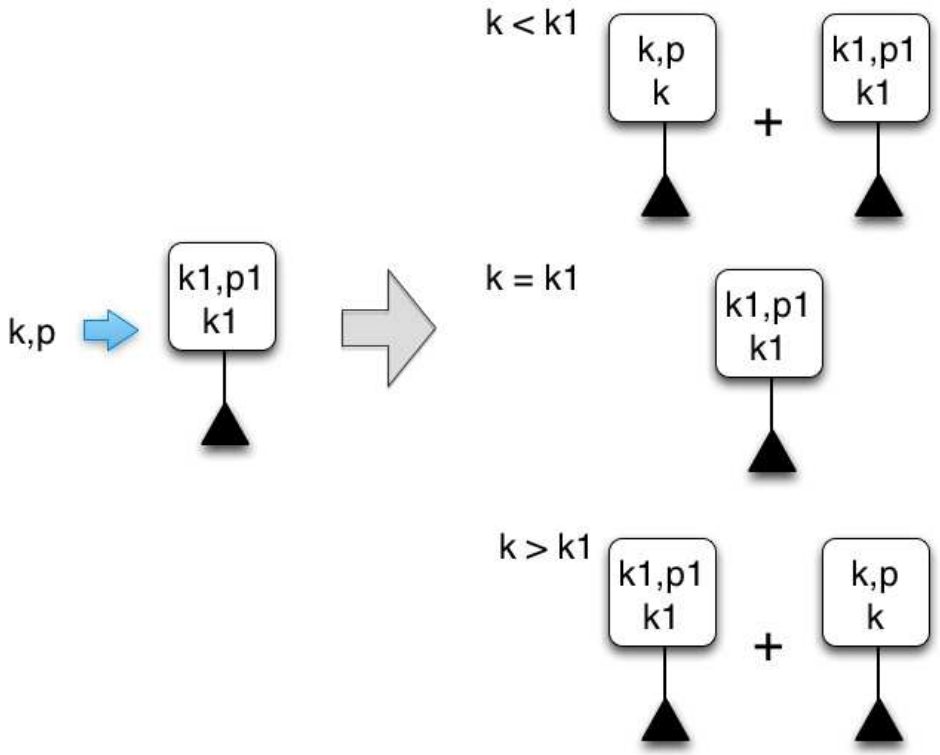
- The *play* operator: +



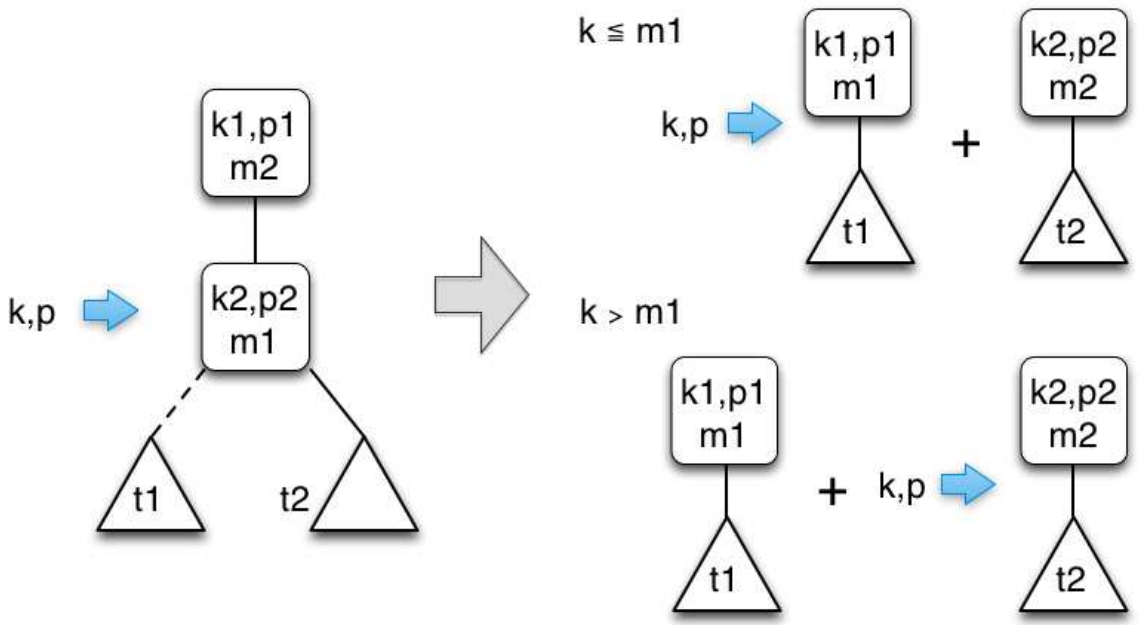
insert (0)



insert (1)



insert (2)


k1,p1
m2
k,p
k2,p2
m1
t1
t2
k ≦ m1
k,p
k1,p1
m1
t1
+
k2,p2
m2
t2
k > m1
k1,p1
m1
t1
+
k,p
k2,p2
m2
t2