Generic Priority Search
generalize everything

• many common algorithms follow a general theme
• BFS uses queue, Dijkstra and Prim a priority queue
• we will refer to a queue like object Q
• methods Q.inject, Q.extract, Q.adjust
• if min PQ, inject=insert, extract=removeMin, adjust=decreaseKey
• we will call the generic PQ a PQoid
generic priority search method

for all nodes \( v \in V \)
  initialize \( v \)
set start node \( s \)

define PQoid \( Q \)
initialize \( Q \)

while \( Q \) not empty
  \( u = Q\).extract
  for each \( v \) such that \((u,v)\) is an edge
    if \( v \in Q \)
      relaxGEN(u,v)
generic relax method

-- needs to be defined for each algorithm

relaxGEN (u,v)
    modify v.key based on u.key
    Q.adjust(v)
breadth first search

Pqoid Q defined as
  set S (vertices not yet seen)
queue F (regular FIFO queue)

Q.insert(v)
  add v to set S

Q.extract(v)
  return F.dequeue

Q.adjust(v)
  remove v from S
  F.enqueue(v)

note: v belongs to Q iff v ∈ S

relaxBFS(u, v)
  v.key = u.key + 1
  Q.adjust(v)

initialize nodes
for all v ∈ V
  v.key = UNDEF
s.key = 0

initialize Q
S = V
Q.adjust(s)

problem here is to take away the "priority" from the priority queue and make it act like a regular queue
prim’s

Pqoid Q is a min PQ
- insert = insert
- extract=removeMin
- adjust=decreaseKey

initialize nodes
for all \( v \in V \)
- \( v.key = \infty \)
- \( s.key = 0 \)

initialize Q
for all \( v \in V \)
- \( Q.insert(v) \)

relaxPrim(u,v)
- \( v.key = \text{MIN} [ v.key, W[u,v] ] \)
- \( Q.adjust(v) \)

\( W[u,v] \) is the weight of edge \((u,v)\)
key means distance from s

PQoid Q is a min PQ
• insert = insert
• extract=removeMin
• adjust=decreaseKey

relaxDIJ(u,v)
  v.key = MIN[ v.key, u.key+W[u,v] ]
  Q.adjust(v)

initialize nodes
for all v∈V
  v.key = ∞
  s.key = 0

initialize Q
for all v∈V
  Q.insert(v)
bandwidth (C-7.7, GT)

key means bandwidth to s

PQoid Q is a maxPQ
- insert = insert
- extract = removeMax
- adjust = increaseKey

initialize nodes
for all v ∈ V
  v.key = 0
s.key = ∞
initialize Q
for all v ∈ V
  Q.insert(v)

relaxBAND(u, v)
  v.key = MAX[ v.key, MIN[u.key, W[u, v]] ]
Q.adjust(v)
generalize everything?

• can’t make Dijkstra’s handle longest paths
• but can many other things
• what is algebraic structure of operations?
• ... closed semirings ...