Lecture notes for "Analysis of Algorithms":

Dynamic All-Pairs Shortest Paths

Lecturer: *Uri Zwick*

December 2010

Function `apsp`$(G = (V, E, c))$

```
t ← 0
foreach u ∈ V do
    E[u] ← ∅
    π[u] ← path(u)
    d[u, u] ← 0
    p[u, u] ← π[u]
foreach u ≠ v ∈ V do
    P[u, v] ← heap()
insert-edges(E)
build-paths()
```

Function `insert-edges`(E_{ins})

```
foreach e = (u, v) ∈ E_ins do
    insert(E[u], e)
    insert(E[v], e)
    π[e] ← path(e)
    heap-insert(P[u, v], π[e], cost[e])
```

Function `delete-edges`(E_{del})

```
foreach e = (u, v) ∈ E_ins do
    delete(E[u], e)
    delete(E[v], e)
    remove-path(π[e])
```

References

[DI04] C. Demetrescu and G.F. Italiano. A new approach to dynamic all pairs shortest paths. *Journal of the ACM*, 51(6):968–992, 2004.

Function `build-paths()`

```
Q ← heap()
init-build-paths()
while Q ≠ φ do
    (u, v) ← extract-min(Q)
    new-shortest-path(p[u, v])
```

Function `init-build-paths()`

```
foreach u ≠ v ∈ V do
    if p[u, v] = null and P[u, v] ≠ φ
    then
        π ← find-min(P[u, v])
        d[u, v] ← cost[π]
        p[u, v] ← π
        heap-insert(Q, (u, v), d[u, v])
```

Function `new-shortest-path(π)`

```
if sel[π] = false then
    sel[π] ← true
    insert(SL[r[π]], π)
    insert(SR[l[π]], π)
    foreach π′ ∈ SL[l[π]] do
        π″ ← path(π′, π)
        examine(π″)
    foreach π′ ∈ SR[r[π]] do
        π″ ← path(π, π′)
        examine(π″)
```

Function `examine(π)`

```
u ← start[π] ; v ← end[π]
heap-insert(P[u, v], π, cost[π])
if cost[π] < d[u, v] then
    d[u, v] ← cost[π]
    if p[u, v] = null then
        heap-insert(Q, (u, v), d[u, v])
    else
        decrease-key(Q, (u, v), d[u, v])
    p[u, v] ← π
```

Figure 1: The functions `build-paths` and `new-shortest-path`.

Function `update(`E_{del}, E_{ins}`)`

```
delete-edges(E_del)
insert-edges(E_ins)
build-paths()
```

Function `vertex-update(`v, E_{new}`)`

```
update(E[v], E_new)
```

Function `full-update(`v, E_{new}`)`

```
t ← t + 1
v[t] ← v
vertex-update(v, E_new)
j ← 1
while j < t do
    dummy-update(v[t − j])
    j ← 2j
```

Function `dummy-update(`v`)`

```
vertex-update(v, E[v])
```

Figure 2: Updating shortest paths.

Function `path`(v)

$\pi \leftarrow$ `new-path`()
$l[\pi] \leftarrow null$
$r[\pi] \leftarrow null$
$start[\pi] \leftarrow v$
$end[\pi] \leftarrow v$
$first[\pi] \leftarrow null$
$last[\pi] \leftarrow null$
$cost[\pi] \leftarrow 0$
$sel[\pi] \leftarrow$ `true`
$GL[\pi], GR[\pi] \leftarrow \emptyset$
$SL[\pi], SR[\pi] \leftarrow \emptyset$
return π

Function `path`$(e = (u, v))$

$\pi \leftarrow$ `new-path`()
$l[\pi] \leftarrow \pi[u]$
$r[\pi] \leftarrow \pi[v]$
$start[\pi] \leftarrow u$
$end[\pi] \leftarrow v$
$first[\pi] \leftarrow e$
$last[\pi] \leftarrow e$
$cost[\pi] \leftarrow c[e]$
$sel[\pi] \leftarrow$ `false`
$GL[\pi], GR[\pi] \leftarrow \emptyset$
$SL[\pi], SR[\pi] \leftarrow \emptyset$
`insert`$(GL[\pi[v]], \pi)$
`insert`$(GR[\pi[u]], \pi)$
return π

Function `path`(π_1, π_2)

if $r[\pi_1] \neq l[\pi_2]$ **then** error
$\pi \leftarrow$ `new-path`()
$l[\pi] \leftarrow \pi_1$
$r[\pi] \leftarrow \pi_2$
$start[\pi] \leftarrow start[\pi_1]$
$end[\pi] \leftarrow end[\pi_2]$
$first[\pi] \leftarrow first[\pi_1]$
$last[\pi] \leftarrow last[\pi_2]$
$cost[\pi] \leftarrow c[first[\pi]] + cost[\pi_2]$
$sel[\pi] \leftarrow$ `false`
$GL[\pi], GR[\pi] \leftarrow \emptyset$
$SL[\pi], SR[\pi] \leftarrow \emptyset$
`insert`$(GL[\pi_2], \pi)$
`insert`$(GR[\pi_1], \pi)$
return π

Figure 3: Generating a new path and inserting it into the path system.

Function `remove-path`(π)

$u \leftarrow start[\pi]$; $v \leftarrow end[\pi]$
if $\pi = p[u, v]$ **then**
  $p[u, v] \leftarrow null$
  $d[u, v] \leftarrow \infty$
`heap-delete`$(P[u, v], \pi)$
if $l[\pi] \neq null$ **then**
  `delete`$(GR[l[\pi]], \pi)$
  `delete`$(GL[r[\pi]], \pi)$
  if $sel[\pi] =$ `true` **then**
    `delete`$(SR[l[\pi]], \pi)$
    `delete`$(SL[r[\pi]], \pi)$
foreach $\pi' \in GL[\pi] \cup GR[\pi]$ **do**
  `remove-path`(π')

Figure 4: Removing a path and all its extensions from the system.