Makeup 3:30–5 on 10/15

1 Fancy Push Relabel Algorithms

1.1 Highest Label

Highest label (more sophisticated fifo):

- idea: avoid sending nonsaturating pushes down a path more than once
- keep vertices arranged by distance label (in buckets)
- always discharge from highest label (flow "accumulates" into fewer piles as moves towards sink)
- easy analysis: if n discharges without relabel, done.
- so 1 relabel every n discharges
- so $O(n^3)$ discharges/nonsaturating pushes.
- so $O(n^3)$ time since relabels, sat pushes $O(nm)$.

Keeping track of level:

- like bucketing shortest paths algorithms
- keep pointer to current highest level
- raise when relabel if necessary
- advance downward to find next nonempty bucket
- total raising $O(n^2)$
- also bound total descent.

Better analysis:

- consider phase between 2 relabels
- each node does only 1 nonsaturating push
- consider inforest of nonsaturating pushes in phase
- decompose into "trajectories" (paths) starting at leaves
- note each leaf must have recieved its excess due to a saturating push
- phase *short* if max forest depth less than $n/\sqrt{m}$, *long* otherwise.
- short phases:
  - short path has $O(n/\sqrt{m})$ nonsat pushes

- each startes with one of $O(nm)$ sat pushes or relabels
- so $O(n^2\sqrt{m})$ total nonsat pushes

- long phases:
  - define *length* of phase to total drop in maximum distance
  - claim: sum of phase lengths $O(n^2)$:
    * decreases must be balanced by increases
    * total increase (relabels) $O(n^2)$
  - number of long phases at most $n^2/(n/\sqrt{m}) = O(n\sqrt{m})$
  - phase has only n pushes
  - so total $O(n^2\sqrt{m})$

Best known strong poly bound for push-relabel without fancy data structs.

1.2 Excess Scaling

Way to achieve $O(nm)$ without data structs, but must discard strong polynomiality.
Basic idea: make sure your pushes send lots of flow.
Instead of highest level, do lowest level!
Can explain by bit shifts, but slightly cleaner to talk about Δ-phases:

- starts with all excesses below Δ
- ends with all excesses below $\Delta/2$
- initially $\Delta = U$
- when $\Delta < 1$, done.
- $O(\log U)$ phases
- each takes $O(nm)$ time
- so $O(nm \log U)$.

Doing a phase: make sure pushes are big

- *large excess* nodes have $e(v) \geq \Delta/2$
- push maximum possible without exceeding Δ excess at destination
- (turns some potentially saturating pushes nonsaturating)
- to ensure big push, always push from large excess with smallest label
- if push nonsaturating, has value at least $\Delta/2$
  - large excess source has at least $\Delta/2$,

- small excess dest can receieve at least this much without going over Δ

Claim: $O(n^2)$ nonsaturating pushes per phase:

- potential function

$$\Phi = \sum d(i)e(i)/\Delta$$

- relabel increases by total of $O((n^2\Delta)/\Delta) = O(n^2)$
- saturating push decreases
- nonsaturating push sense $\Delta/2$ downhill: decrease by $1/2$
- so $O(n^2)$ nonsaturating pushes.
- note: in this alg, *saturating pushes* form bottleneck.

Deduce: $O(nm + n^2 \log U)$ running time.

1.3 Wrapup

Text discusses practical choice, argues for:

- shortest aug path simple, often good enough
- highest label best in practice if time to code
- excess scaling also good.

Open: $O(nm)$-ish without scaling, data structs

2 Min-Cost Flow

Many different max-flows in a graph. How compare?

- *cost* $c(e)$ to send a unit of flow on edge e
- find max-flow minimizing $\sum c(e)f(e)$
- costs may be positive or negative!
- note: pushing flow on cost c edge create residual cost $-c$ edge.
- also easy to find min-cost flow of given value v less than max (add bottleneck source edge of capacity v)

Clearly, generalizes max-flow. Also shortest path:

- How send flow 1 unit of flow?

- just use shortest path
- more generally, flow decompose into paths and cycles
- cost of flow is sum of costs of paths and cycles.
- each path costs at most nC (C = max cost)
- cost of flow at most mUC

Min-cost circulation:

- no source or sink
- just find flow satisfying balance everywhere, min-cost
- if satisfy balance everywhere, all flow must be going in circles!
- more formally: circulation can be decomposed into just cycles.
- hard to define in max-flow perspective, but makes sense once allow negative cost arcs.
- reduction to min-cost flow: add disconnected s, t.
- reduction from min-cost flow:
  - add s-t arc of "infinite" capacity, "infinite" negative cost
  - of course, circulation will push max possible through this edge
  - how much can it? max s-t flow
  - so of course, suff to assign capacity equal to max-flow value
  - see later, sufficient to assign cost $-nU$ (good for scaling)
- another reduction from min-cost flow:
  - find any old max-flow f
  - consider min-cost flow f^*
  - difference $f^* - f$ is a circulation (note: diff of two equal capacity flows is a circulation)
  - so find circulation q in G_f.
  - $q + f$ is a flow in G (note: flow+circulation=flow of same capacity)
  - cost is $c(q) + c(f)$
  - so adding min-cost q in G_f yields min-cost flow

Deciding optimality:

- given a max-flow. How decide optimal?
- by above, optimal if min-cost residual circulation is 0

- suppose not. so have negative cost circulation
- decomposes into cycles of flow
- one must have negative cost.
- so, if f nonoptimal, negative cost cycles in G_f
- converse too: if negative cost cycle, have negative cost circulation. So min-cost< 0.