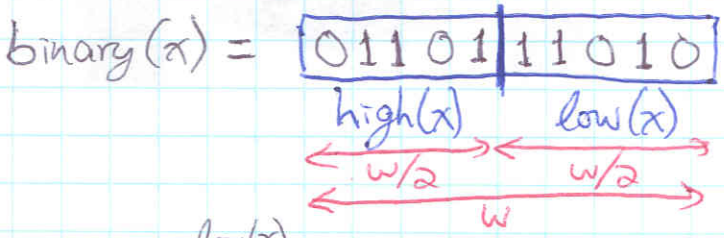
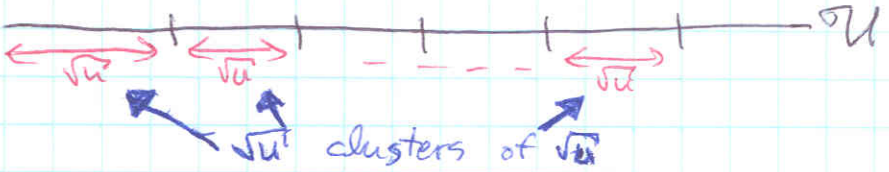
van Emde Boas (vEB)

6.897
L 9.3
Mar. 3, 2005

- $O(\lg w)$ time means we "binary search" on bits of query word x



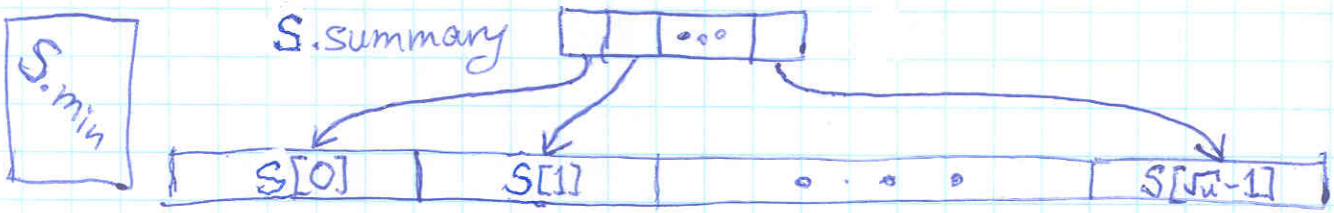
- low-order bits $\overset{\text{low}(x)}{\wedge}$ distinguish $2^{w/2} = \sqrt{u}$ consecutive items
- high-order bits high(x) distinguish $2^{w/2} = \sqrt{u}$ of these clusters:



- vEB structure follows this recursion:

vEB structure (S) of size u =

- substructures $S[0], S[1], \ldots, S[\sqrt{u}-1]$ each of size $\sqrt{u}$ representing partition of universe (above)
- substructure S.summary of size $\sqrt{u}$ where $x \in S.\text{summary} \Leftrightarrow S[x]$ is nonempty
- minimum element S.min
  - not stored recursively in an $S[i]$ (but still counts as making S nonempty)
- $S.\text{min} = \text{none} \Leftrightarrow S$ is empty



- also augment to store S.max (stored recursively)

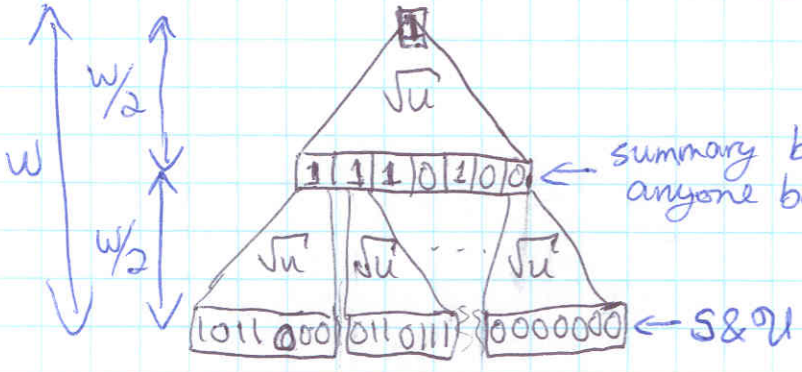
6.897
L 9.4
Mar. 3, 2005

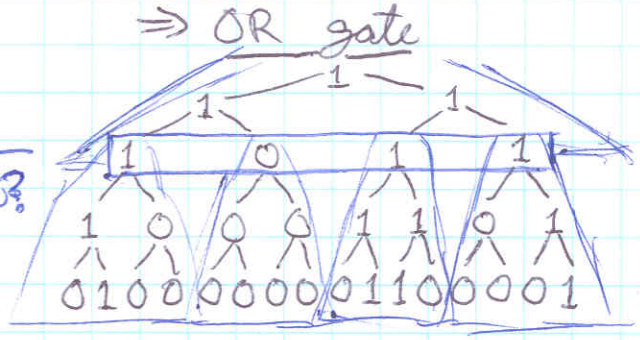
Successor(x, S):
- if $x < S.\min$: return $S.\min$
- x is at position $\text{low}(x)$ in $S[\text{high}(x)]$
- if $\text{low}(x) < S[\text{high}(x)].\max$ ← key: augmentation
  then: return $\text{high}(x) \cdot \sqrt{|S|} + \text{Successor}(\text{low}(x), S[\text{high}(x)])$
  else: $i \leftarrow \text{Successor}(\text{high}(x), S.\text{summary})$
  return $i \cdot \sqrt{|S|} + S[i].\min$ ←

Insert(x, S):
- if $S.\min = \text{none}$ then $S.\min \leftarrow x$
  stop
- if $S[\text{high}(x)]$ is empty $(.\min = \text{none})$
  then: $S[\text{high}(x)].\min \leftarrow \text{low}(x)$ ← key: not stored recursively
  $\text{Insert}(\text{high}(x), S.\text{summary})$
  else: $\text{Insert}(\text{low}(x), S[\text{high}(x)])$

Analysis:
- key: one recursive call on word half the size
  ⇒ $O(\lg w)$
- Predecessor & Delete similar

Expand recursion: tree view





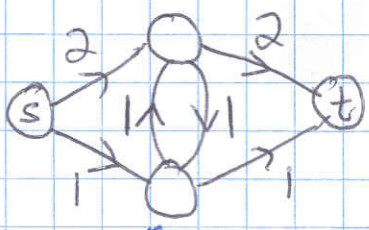
- can't afford to update all these bits ~
  but the ones really needed by Successor are there
- not storing min recursively avoids a lot of these updates

Network Flow / Maximum Flow

CONCISE VERSION WITHOUT RAW/POSITIVE FLOWS (JUST NET FLOWS)

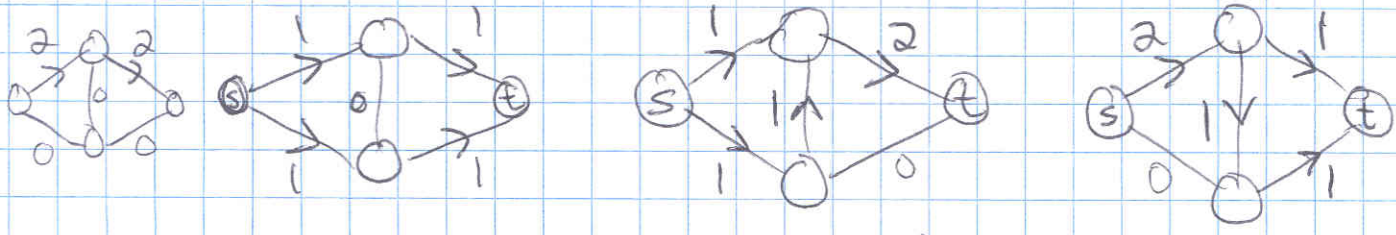
6.854
L6.1
Sept. 24, 2003

Network:
- directed graph
- source vertex s
- sink vertex t
- capacity $u(e) = u(v,w) > 0$ for each directed edge $e = (v,w)$
  <<"u", not "c" — "c" used for cost later>>
- $u(v,w) = 0$ if (v,w) not an edge



Treat as one (bidirectional) edge in flow — can use just one

Flow: real number $f(v,w)$ $\forall v,w$ (nonzero only if (v,w) or (w,v) an edge) satisfying:
- skew symmetry: $f(v,w) = -f(w,v)$
- conservation: $\sum_w f(v,w) = 0$ unless $v = s$ or t
- capacity: $f(v,w) \le u(v,w)$ (flow is feasible)



Value of flow f: $|f| = \sum_w f(s,w)$

Max-flow problem: find flow of maximum value

Intuition: $f(e)$ is a rate, not quantity
(so don't have to worry about startup delays etc.)

E.g.: water pipes,
routing commodities,
messages under bandwidth constraints, ...

6.854
L6.2
Sept. 24, 2003

Residual network G_f of flow f in network G
- capacities $u'(v,w) = u(v,w) - f(v,w)$
- f feasible $\Rightarrow$ $u'(v,w) \geq 0$ & $\leq 2 \cdot u(v,w)$ [$f(v,w)$ might be negative]
- G_f has edges (v,w) whenever $u'(v,w) > 0$.

MAX FLOW IN G_f = MAX FLOW IN G (JUST ADD/SUB f)

Augmenting path is a directed path from s to t in residual network
- can push additional flow along such a path in original network G:
$$\min\{u'(s,v_1), u'(v_1,v_2), \ldots, u'(v_n,t)\}$$
- augmenting path exists $\Rightarrow$ not max. flow
- no augmenting path $\Rightarrow$ max flow??

MORE GENERALLY: CAN ADD ANY FLOW IN G_f TO FLOW IN G

Cut is a partition of vertices into 2 groups: S & $\bar{S} = V \setminus S$.
s-t cut has $s \in S$ and $t \in \bar{S}$.

All s-t cuts carry same flow:
- net flow along cut S: $f(S) = \sum_{v \in S} \sum_{w \in \bar{S}} f(v,w) =: f(S,\bar{S})$
- claim: $f(S) = |f|$ for all s-t cuts $S, \bar{S}$.
  - true for $\{s\}, V \setminus \{s\}$.
  - if we move v from $\bar{S}$ to S then add $f(v,\bar{S})$ & subtract $f(S,v) = -f(v,S)$ } really $S - \{v\}$, $\bar{S} - \{v\}$
  - total change: $f(v,\bar{S}) + f(v,S) = f(v, \underbrace{V}_{S \cup \bar{S}}) = 0$ (conservation)

Capacity/Value of cut: $u(S) = u(S,\bar{S}) = \sum_{v \in S} \sum_{w \in \bar{S}} u(v,w)$

value of any flow = net flow along any s-t cut ≤ value of cut

$|f| = f(S) \leq u(S)$

$\Rightarrow |\text{max flow}| \leq u(\text{min } s\text{-}t \text{ cut})$

Max-flow min-cut theorem: $|\text{max flow}| = u(\text{min } s\text{-}t \text{ cut})$

foreshadowing more general duality in LP

- max flow has no augmenting paths
- $\Rightarrow$ residual graph G_f is s-t disconnected
- $\Rightarrow$ $S = \{$vertices reachable from s in $G_f\}$ is an s-t cut
- all edges (v,w) leaving S have 0 residual
  $\Rightarrow u(v,w) = f(v,w)$

$\Rightarrow u(S) = f(S) = |f|$

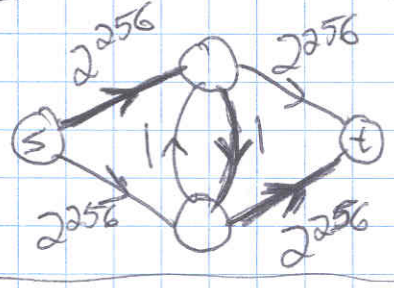
$\Rightarrow$ s-t cut S has value $|f| = |\text{max flow}|$

$\Rightarrow u(\text{min } s\text{-}t \text{ cut}) \leq |\text{max flow}|$ □

Ford-Fulkerson max-flow algorithm

- start with $f(v,w) = 0$
- DFS to find an augmenting path
- augment flow
- repeat until DFS fails

$\Theta(256)$ bits / $\Theta(2^{256})$ augment



Running time:

- if capacities are integral: $O(n \cdot |f|) = O(nmU)$ where $U = \max u$
  - INTEGRALITY
  - weakly polynomial: polynomial in unary encoding of input numbers
- if capacities are rational: still terminates but slower
- if capacities are real: may not terminate

<<What if we choose a good augmenting path? >>

6.854
L6.4
Sept. 24, 2003

(UNCOVERED – DELAYED TO L7)

Bit scaling

- shift in one bit of capacities (high-order first)
- run naive augmenting paths
- repeat

Analysis:

- before next bit shift, some cut has residual capacity 0
- after next bit shift, that cut has ~~capacity~~ residual capacity $\leq m$
  - shifting left 1 bit ($\times 2$) has no effect
  - introducing new low-order bit $\Rightarrow$ +0 or +1.
  - at most m edges straddle cut

$\therefore \leq m$ augmentations restore max flow

- $\lg U$ iterations ($U = \max u$)

Total time: $O(m^2 \lg U)$

- polynomial in bit complexity of input
- not strongly polynomial i.e. polynomial in combinatorial complexity m, n

6.854
L6.5
Sept. 24, 2003

Maximum-capacity augmenting path

- choose augmenting path that improves flow value most
  - can find this path in $O(m \lg n)$ time
    using Dijkstra-like algorithm (ignore cycles)

Lemma: can decompose any flow into $\leq m$ paths with values
- follow any path of positive net flow from s to t
  (once started, can always continue to extend path by conservation)
- anti-augment: subtract augmenting path } add to set of paths
  of maximum possible value
- at least one edge hits 0 flow
- repeat $\leq m$ times because $\leq m$ edges can hit 0 flow
- remaining cycles of flow $\square$

similarly find

- apply lemma to max flow in residual graph (discarding cycles)
  $\Rightarrow$ set of m candidate augmenting paths
  whose total value = total residual flow
- at least one of these paths carries $\geq 1/m$ fraction of residual flow
  $\Rightarrow$ max-capacity augmenting path reduces $|f|$ to $\leq (1-1/m)|f|$
- after $m \ln f$ iterations, $(1-1/m)^{m \ln f} \cdot f \sim (1/e)^{\ln f} \cdot f = 1$
  $\Rightarrow$ total residual capacities in $[0,1]$
- if capacities are integral, one more step suffices.
  $\Rightarrow \leq m \ln f$ augmentations
  $\Rightarrow O(m^2 \ln f \lg n)$ time — also polynomial but not strongly polynomial
- also works for rational capacities