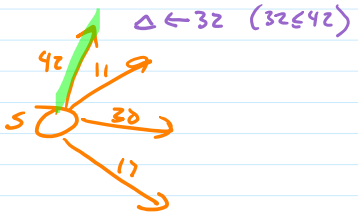
Maximum Flow in Polynomial Time

MAX-FLOW (G)

$f(u,v) \leftarrow 0$ for all $(u,v) \in E$

$G_r \leftarrow G$

$\Delta \leftarrow 2^{\lfloor \log_2 \max_{(s,v)\in E} c(s,v) \rfloor}$



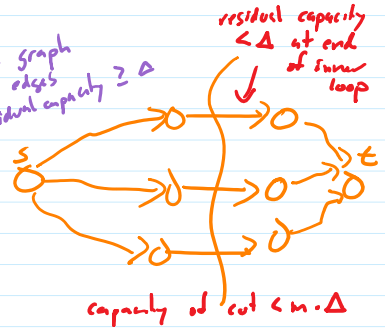
$O(\log_2 C)$ while $\Delta \geq 1$

iterations??? → while there is a path P $s \rightsquigarrow t$ in $G_{r,\Delta}$ ← residual graph w/ only edges w/ residual capacity $\geq \Delta$

$O(n)$ [augment (P, f)
update (G_r, P)

$\Delta \leftarrow \Delta/2$

$O(m)$



total $O(\log_2 C \cdot m \cdot n)$

polynomial in size of graph + # bits in capacities

INV (outermost loop): there is a cut (A, B) s.t. $c(A,B) \leq v(f) + 2m\cdot\Delta$

Basis:

Maintenance: Let $A^* = \{v \mid s \rightsquigarrow v \text{ in } G_{r,\Delta}\}$, $B^* = V - A^*$

$v(f) = f^{out}(A^*) - f^{in}(A^*)$ THM of Mar 27

$= \sum_{\substack{(u,v)\in E \\ u\in A^* \\ v\in B^*}} f(u,v) - \sum_{\substack{(u,v)\in E \\ u\in B^* \\ v\in A^*}} f(u,v)$ def f^{out}, f^{in}

for edges $A^* \to B^*$
$c_r(u,v) = c(u,v) - f(u,v) < \Delta$
(if $c_r(u,v) > \Delta$ then $(u,v) \in G_{r,\Delta}$ so $s \rightsquigarrow u \to v$ ⇒⇐)

$\geq \sum_{\substack{(u,v)\in E \\ u\in A^* \\ v\in B^*}} (c(u,v) - \Delta_{old}) - \sum_{\substack{(u,v)\in E \\ u\in B^* \\ v\in A^*}} \Delta_{old}$

for edges $B^* \to A^*$
$f(u,v) < \Delta$
(if $f(u,v) \geq \Delta$ then $c_r(v,u) \geq \Delta$ so $(v,u) \in G_{r,\Delta}$ and $s \rightsquigarrow v \to u$ ⇒⇐)

$\geq c(A^*, B^*) - m\Delta_{old}$

$= c(A^*, B^*) - 2m\Delta_{new}$

any flow ≤ capacity of any cut ↓ INV ↓

$v(f_{new}) \leq v(f^*) \leq c(A^*, B^*) \leq v(f_{old}) + 2m\Delta$

$v(f_{new}) - v(f_{old}) \leq 2m\Delta$

$m=10$ $\Delta=16$ 320

we add at most this during iteration of outer loop

each iteration of inner adds $\geq \Delta$ flow

so $\leq 2m$ iterations of inner

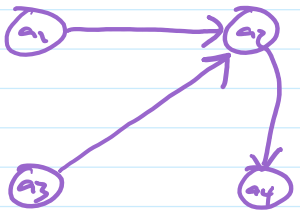
Comparison Graphs

PRE: n ≥ 1, A distinct

```
MAX(A, n)
    max ← A[1]
    for i = 2 to n
        if A[i] > max
            max ← A[i]
    return max
```

n-1 comparisons

Comparison Graph
captures results of comparisons done in alg
small → big



lower bound for MAX: n-1 comparisons

- every alg that solves MAX has worst-case of ≥ n-1 comparisons

to know max, must be path from to what the max is

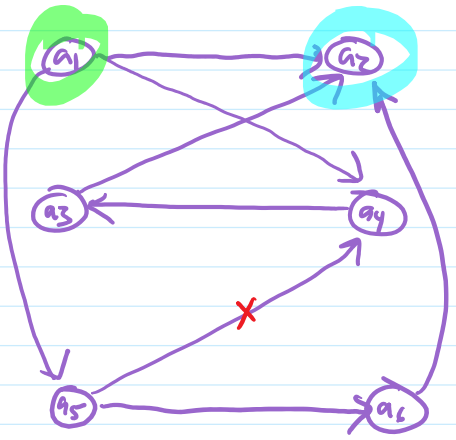
so ≥ n-1 edges

```
MIN_AND_MAX(A, n)
    max ← A[1]
    min ← A[1]
    for i = 2 to n
        if A[i] > max
            max ← A[i]
        else if A[i] < min
            min ← A[i]
    return (min, max)
```

worst case 2n - 2

```
maxc ← []
minc ← []
for i = 1 to n by 2                          n/2
    if A[i] < A[i+1]
        minc.append(A[i])
        maxc.append(A[i+1])
    else
        :
return (MIN(minc), MAX(maxc))
        n/2 - 1   n/2 - 1
```

total $\frac{3n}{2} - 2$



For any MIN_AND_MAX Alg, can find path w/ ≥ $\frac{3n}{2} - 2$ comps

! ∪ unknown
X conds for max
a1 & min

: U unknown
X cands for max
N for min
O ruled out

if comp U vs U choose branch arbitrarily
X vs N follow X bigger branch
X vs X
N vs N arbitrary
X vs O follow X bigger
N vs O follow O bigger
O vs O follow consistent w/ prev

at beg, must have 1 X, 1 N, 0 U, n-2 O

need $\frac{3n}{2} - 2$ comparisons to go from

n U $\longrightarrow$ 1X, 1N, 0U, n-2 O

so any alg for MIN_AND_MAX has worst case $\geq \frac{3n}{2} - 2$

Lower Bound for MIN_AND_MAX is $\frac{3n}{2} - 2$