CSE 589 Part III

The computer is useless
It can only answer questions.

-- Pablo Picasso

Dynamic Programming Summary

1. Formulate answer as recurrence relation or recursive algorithm
2. Show that number of values of recurrence bounded by poly
3. Specify order of evaluation for recurrence so have partial results when you need them.

DP works particularly well for optimization problems with inherent left to right ordering among elements.

Resulting global optimum often much better than solution found with heuristics.

Once you understand it, usually easier to work out from scratch a DP solution than to look it up.

TCP acknowledgement dynamic delay

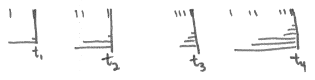
given arrival times $a_1, a_2, \dots, a_n$

Find set of times $t_1, t_2, \dots, t_k$ at which acknowledgements are sent

s.t. $k + \sum_{i=1}^{k} \underbrace{\text{latency for subsequence } j}$ is minimized

$t_k \geq a_n$
$k \geq 1$

$$\sum_{t_{j-1} < a_i \leq t_j} (t_j - a_i)$$



$OPT(i)$ = optimal set of times for $a_1, \dots, a_i$

$OPT(0) =$

$OPT(1) =$

$OPT(i) =$

Country whose coins are minted with denominations $d_1, d_2, \dots, d_k$

Compute min # of coins needed to make change for n units

Readings

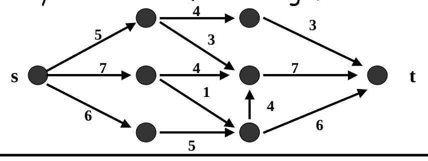
Max Flow

- Skiena, Section 8.4.9
- CLR, chapter 27
- Network Flows, Theory, Algorithms and Applications, by Ahuja, Magnanti and Orlin, chapter 6 ++

Maximum Flow

Input description: a graph (network) G where each edge (v,w) has associated capacity c(v,w), and a specified source node s and sink node t

Problem description: What is the maximum flow you can route from s to t while respecting the capacity constraint of each edge?



Max-flow outline:

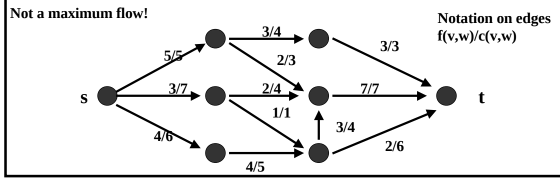
properties of flow
augmenting paths
max-flow min-cut theorem
Ford-Fulkerson method
Edmonds-Karp method
applications
variants: min cost max flow

Properties of Flow: f(v,w) -- flow on edge (v,w)

0<= f(v,w) <= c(v,w) : the flow through an edge cannot exceed the capacity of an edge

for all v except s,t $\Sigma_u f(u,v) = \Sigma_w f(v,w)$: the total flow entering a vertex is equal to total flow exiting vertex

total flow leaving s = total flow entering t.

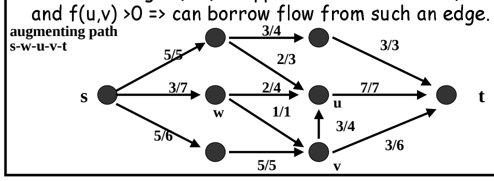


An augmenting path with respect to a given flow f is a

Directed path from s to t which consists of edges from G, but **not necessarily in same direction**. Each edge on path is either:

forward edge: (u,v) in same direction as G and f(u,v) < c(u,v). (c(u,v)-f(u,v) called slack) => has room for more flow.

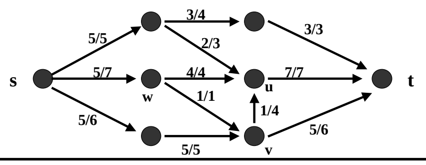
backward edge: (u,v) in opposite direction in G (i.e., (v,u) in E) and f(u,v) >0 => can borrow flow from such an edge.



Using an augmenting path to increase flow

If all edges are forward => move flow through all of them. amount of flow can push = minimum slack

Otherwise, push flow forward on forward edges, deduct flow from backward edges.



Augmenting Path Theorem:

A flow f is maximum iff it admits no augmenting path

Already saw that if flow admits an augmenting path, then it is not maximum.

Suppose f admits no augmenting path. Need to show f maximum.

Cut -- a set of edges that separate s from t.

Capacity of a cut = sum of capacities of edges in cut.

Prove theorem by showing that there is a cut whose capacity = f.

A = vertices s.t. for each v in A, there is augmenting path from s to v. => defines a cut.

Claim: for all edges in cut, f(v,w)=c(v,w)

=> value of flow = capacity of cut defined by A => maximum.

=> Celebrated Max-flow Min-Cut Theorem

The value of a maximum flow in a network is equal to the minimum capacity of a cut.

The Integral Flow Theorem

If the capacities of all edges in the network are integers, then there is a maximum flow whose value is an integer.

Residual Graph w.r.t. flow f

R=(V,E')

an edge (v,w) in E' if either

it is forward edge

- capacity = c(v,w)-f(v,w)

or it is a backward edge

- capacity = f(v,w).

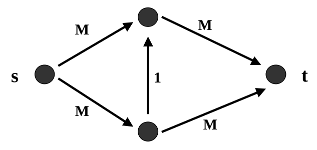
(assuming edges in only one direction)

An augmenting path = a regular directed path from s to t in the residual graph.

Ford-Fulkerson Method (G,s,t)

```
Initialize flow on all edges to 0.
While there is a path p from s to t in residual network R
    δ = minimum capacity along p in R
    augment δ units of flow along p and update R
```

Running Time?



Edmonds-Karp

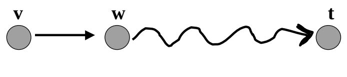
If implement computation of augmenting path using breadth-first search, i.e., if augmenting path is a **shortest path** from s to t in residual network, then the number of augmentations O(|V||E|).

- The shortest-path distance from v to t in R is non-decreasing.
- The number of times an edge (u,v) can become **critical**, i.e., residual capacity of (u,v) = residual capacity of augmenting path, is O(|V|).

The shortest path distance from v to t in R_f is non-decreasing.

Proof by contradiction:

Consider first time SP (in residual graph) to t decreases from some node v. Let v be the closest such node to t. SP in f' from v to t begins with edge (v,w)



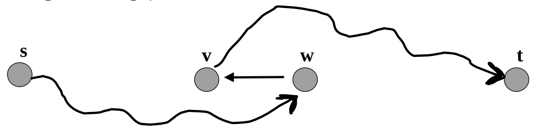
d'(v)=d'(w)+ 1.

Claim: f(v,w)=c(v,w). If not (v,w) in R_f

d(v) <= d(w)+1 <= d'(w)+1 = d'(v). Contradiction.

Shortest paths non-decreasing, cont.

Therefore (w,v) in R_f, and in order for (v,w) in $R_{f'}$, augmenting path must be:



=> d(v) = d(w)-1 <= d'(w)-1 = d'(v)-2

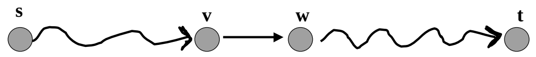
=> d(v) <= d'(v).

Contradiction.

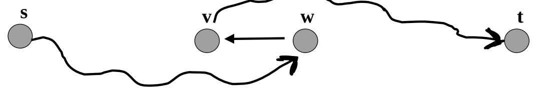
✓

Lemma: between any two consecutive saturations of (v,w), both d(v) and d(w) increase by at least 2.

In first saturation, increasing flow along augmenting path, and d(v)= d(w)+1.



Before edge can be saturated again, must send flow back from w to v. => d′(w)= d′(v)+1.



In next saturation, again have d″(v)=d″(w)+ 1.
=> d″(v)= d″(w)+1 >= d′(w)+1 >= d′(v)+2 >= d(v)+2.

=> Running time of Edmonds-Karp is $O(m^2n)$

Corollary 1: An edge (v,w) can be saturated at most O(n) times.

Corollary 2: The total # of flow augmentations performed is O(mn)

Since each augmentation can be performed in O(m) time => $O(m^2n)$ time.

Can be improved to nearly O(mn) with super-fancy data structures.

General notion of residual graph

implicit in earlier discussion:
edge in only one direction
(u,v) or (v,u) but not both

in general, can have both

$$\begin{tikzcd} v \arrow[r, shift left, "c(v,w)"] & w \arrow[l, shift left, "c(w,v)"] \end{tikzcd}$$

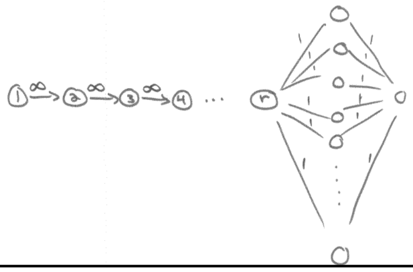
$$\begin{tikzcd} v \arrow[r, shift left, "4/6"] & w \arrow[l, shift left, "7/10"] \end{tikzcd} \xRightarrow[\text{so one direction 0}]{\text{always cancel out}} \begin{tikzcd} v \arrow[r, shift left, "0"] & w \arrow[l, shift left, "3"] \end{tikzcd}$$

How to define residual?

$r(w,v) = 7$ $\quad\quad$ $r(v,w) = 6+3 = 9$

$$r(v,w) = \underbrace{c(v,w) - f(v,w)}_{\text{add flow up to capacity}} + \underbrace{f(w,v)}_{\text{cancel flow in other direction}}$$

drawback of augmenting path algs is computationally expensive operation of sending flow along a path, which takes O(n) time



Fastest max-flow algorithms: preflow-push

Flood the network so that some nodes have excess or buildup of flow

algorithms incrementally relieve flow from nodes with excess by sending flow towards sink or back towards source.

Some applications of max-flow and max-flow min-cut theorem

Scheduling on uniform parallel machines
Bipartite matching
Network connectivity
Video on demand

✔

Scheduling on uniform parallel machines

Have a set J of jobs, M uniform parallel machines.

Each job j has processing requirement p_j, release date r_j and due date d_j. $d_j >= r_j + p_j$

A machine can work on only one job at a time and each job can be processed by at most one machine at a time, but preemptions are allowed.

Scheduling problem: determine a feasible schedule that completes all jobs before their due dates or show that no such schedule exists.

Scheduling on M uniform parallel machines

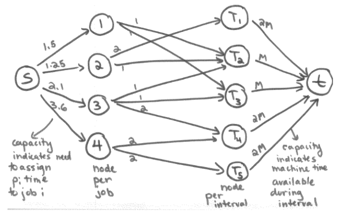
Example

j	1	2	3	4
p_j	1.5	1.25	2.1	3.6
r_j	3	1	3	5
d_j	5	4	7	9

rank release & due dates in ascending order

$$\underbrace{1 \quad 3}_{T_1}\ \underbrace{3\ \ 4}_{T_2}\ \underbrace{4\ \ 5}_{T_3}\ \underbrace{5 \quad 7}_{T_4}\ \underbrace{7 \quad 9}_{T_5}$$

within interval set of jobs released & due doesn't change


s
1.5
1.25
2.1
3.6
1
2
3
4
T1
T2
T3
T4
T5
t
2M
M
capacity indicates need to assign pj time to job j
node per job
node per interval
capacity indicates machine time available during interval

edge from job j to interval T_i if j released before that interval and due after.

capacity = machine time that can be allocated to job during time

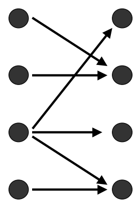
Solve max flow problem

flow = Σp_j <=> feasible schedule

Bipartite Matching

Input: a bipartite graph G=(V,E)

Output: Largest-size set of edges M from E such that each vertex in V is incident to at most one edge of S



Network Connectivity

What is the minimum number of links in the network such that if that many links go down, it is possible for nodes s and t to become disconnected?

What is the minimum number of nodes in the network such that if that many nodes go down, it is possible for nodes s and t to become disconnected?

Video on Demand

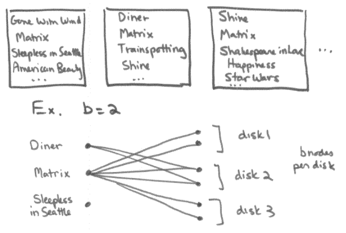
M storage devices (e.g., disks), each capable of supporting b simultaneous streams.

k movies, one copy of each on 3 disks.

Given set of R movie requests, how would you assign the requests to disks so that no disk is assigned more than b requests and the maximum number of requests is served?

Video on demand

k movies, M disks (each has bandwidth b)
some movies stored on multiple disks

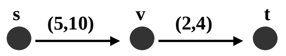


Find maximum matching

Other network flow problems: 1. With lower bounds on flow.

For each (v,w): $0 \le lb(v,w) \le f(v,w) \le c(v,w)$

Not always possible:



Other network flow problems: 2. Minimum flow

Want to send minimum amount of flow from source to sink, while satisfying certain lower and upper bounds on flow on each edge.

Other network flow problems: 3. Min-cost max-flow

Input description: a graph (network) G where each edge (v,w) has associated capacity c(v,w), and **a cost *cost*(v,w)**.

Problem description: What is the maximum flow **of minimum cost** you can route from s to t while respecting the capacity constraint of each edge?

The cost of a flow :

$$\Sigma_{f(v,w)>0}\ \mathit{cost}(v,w)f(v,w)$$

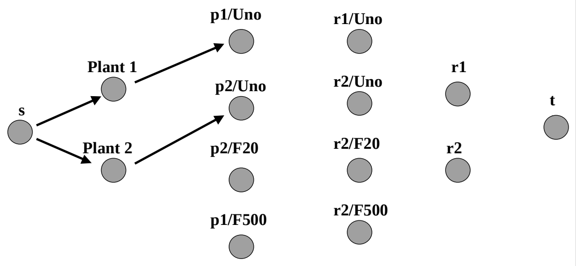
Classical application: Transportation Problem

Firm has p plants with known supplies, q warehouses with known demands.

Want to identify flow that satisfies the demands at warehouses from available supplies at plants and minimizes shipping costs.

Min cost flow yields an optimal production and shipping schedule.

Example: Fiat makes Uno and Ferrari's.

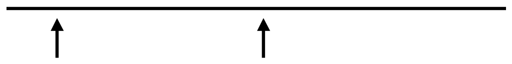


cost	0	production cost	shipping cost	0	0
cap	supply of plant	max production	capacity imposed by dist'n channel	model demand	Retailer demand

✖

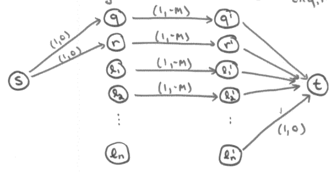
Disk head scheduling

Have disk with 2 heads, and sequence of requests to read data on disk at locations $l_1, \ldots, l_n$.
How to schedule movement of disk heads so as to minimize total head movement?



Disk head scheduling $\ell_1, \ell_2, \ldots, \ell_n$
cost of moving head between locations

start on q, r



M very large positive #

edges from

$q' \longrightarrow \ell_i$ $\quad \forall\ 1 \le i \le m$
cap=1 $\quad$ cost = dist (q', ℓ_i)

$r' \longrightarrow \ell_i$ $\quad \forall\ 1 \le i \le m$
cap=1 $\quad$ cost = dist (r', ℓ_i)

$\ell_i' \longrightarrow \ell_j$ $\quad j > i$
cap=1 $\quad$ cost = dist (ℓ_i, ℓ_j)

✗