CS 6381: Lecture #1 8.12.2020

Graph Algorithms & Combinatorial Optimization

You will find material at:
www.utdallas.edu/~chandra
under CS 6381. as well as on elearning site and its Blackboard-Collaborate.

We will meet regularly in Course room of e-learning ~~blackboad~~ Blackboard collaborate on
TR: 10:00 - 11:15

"Printed" Notes at my home page: www.utdallas.edu/~chandra

Audio/Video: Versions of pdf: my home page as well as e-learning; Recorded Versions at e-learning; (I am trying to post these on Microsoft Teams as well for the purpose of ADA)

Office Hours: ☒ Information on my webpage.
www.utdallas.~~edu~~/~chandra
Most likely: M: 1:30-3:00

Prerequisite CS 6363

Your part: Several HW Assignments, one take home exam; Presentation on an agreed upon topic.

List of topics to be covered : my home page.

In CS6363 : Topics from Graph Algorithms Covered:

Spanning Tree Problem (Undirected graphs)
Ch 23 CLRS
linear, 5 algorithms (Greedy), Imp.

Shortestpath : a) Single Origin (directed graphs)
3 algorithms (DP)

(b) All pairs
2 algorithms (DP)

Maximum Flow : Single-origin, Single destination
2 algorithms (Improvement)
Directed graph.
(Linear Prog. Duality)
Max-Flow-Min-Cut Theorem

Postponed in CS6363 to CS6381 [Promises made]

1. Pf of Correctness of Improvement algorithm for Minimum Spanning Trees.
2. Finding an s-t shortest path in undir. graph with possible negative edge weights but no negative Undirected cycle [Not based on DP!]
3. Ratio Spanning Trees
4. A counter example in Max Flow algorithm of Ford. Fulkerson.

5. Complete proof of Edmonds-Karp algorithm for Max-Flow; its complexity.

Review of Maximum-Flow (Single origin. Single destination)

Input: A directed (simple) graph $G=[V;E]$ with edge capacities $c(u,v) \geq 0 \ \forall (u,v) \in E$; An origin $s \in V$ and a destination $t \in V$. [without loss, assume $s \neq t$].

Want: Maximum feasible flow F from s to t in G

Def: a feasible flow: $f(u,v)$; $(u,v) \in E$; with

$$0 \leq f(u,v) \leq c(u,v) \ \forall (u,v) \in E.$$

(Capacity constraints)

Formulation I — Node-edge Formulation

And:

$$\sum_{\substack{v \in V \\ (u,v) \in E}} f(u,v) - \sum_{\substack{v \in V \\ (v,u) \in E}} f(v,u) = \begin{cases} F & u = s \\ 0 & u \neq s,t \\ -F & u = t \end{cases}$$

(Conservation constraints)

Problem Max F subject to above constraints.

Path-edge formulation:

Let $\mathcal{P}$ be the set of (directed) paths from s to t in G.

Let $x_P, P \in \mathcal{P}$ represent "flow" along path P.

Feasibility: $x_P \geq 0 \ \forall P \in \mathcal{P}$

$$\sum_{P: e \in P} x_P \leq c(e) \quad e \in E.$$

$$\text{Max} \sum_{P \in \mathcal{P}} x_P$$

Node-edge formulation is Compact (polynomial # of variables and polynomial number of constraints)

Path-edge formulation is not Compact since # of s-t paths could be exponential. But it is useful later on for more complicated problems.

Equivalence: (1) Path-edge $\longrightarrow$ Node-edge

Given x_p, $P \in \mathcal{P}$, let $f(u,v) = \sum_{P:(u,v)\in P} x_p$

Since $x_p \geq 0$, for a feasible flow, $f(u,v) \geq 0$.

Since $\sum_{P:(u,v)\in P} x_p \leq c(u,v) \quad \forall (u,v) \in E$,

$f(u,v) \leq c(u,v) \quad \forall (u,v) \in E$.

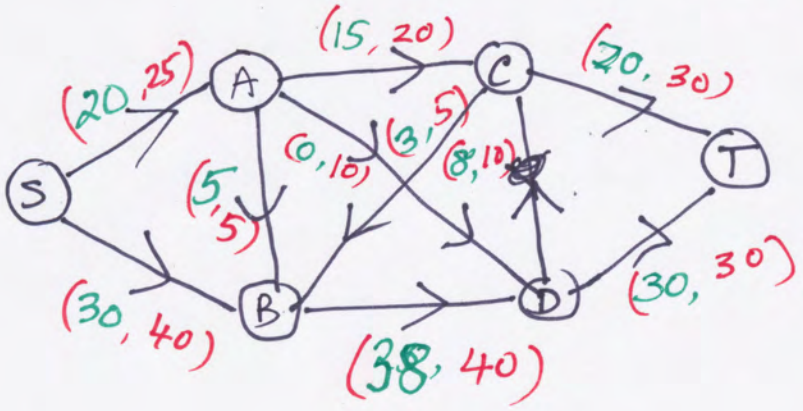
Since x_p satisfies conservation (by definition)

$f(u,v)$ also " "

$\sum_{P\in\mathcal{P}} x_p = F$

Going in the reverse direction needs a "Path-decomposition" of a feasible flow type algorithm. This is shown via an example in the next page

Consider the following example: Red: Capacities
Green: flow.



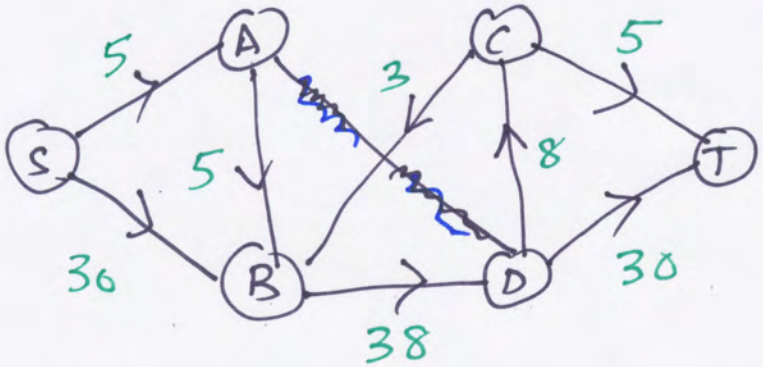
(S,A)	20, 25
(S,B)	30, 40
(A,B)	5, 5
(A,D)	0, 10
(A,C)	15, 20
(B D)	38, 40
(C B)	3, 5
(C,T)	20, 30
(D,C)	8, 10
(D,T)	30, 30

Using only edges that have positive flows, try (using XFS) a directed path from s, to t [with all edges having positive flows] and assign to that path, the minimum of edge flows on that path.

Ex: S-A-C-T: 15

Reduce all flows on edges along this path by 15.

Remaining flows:

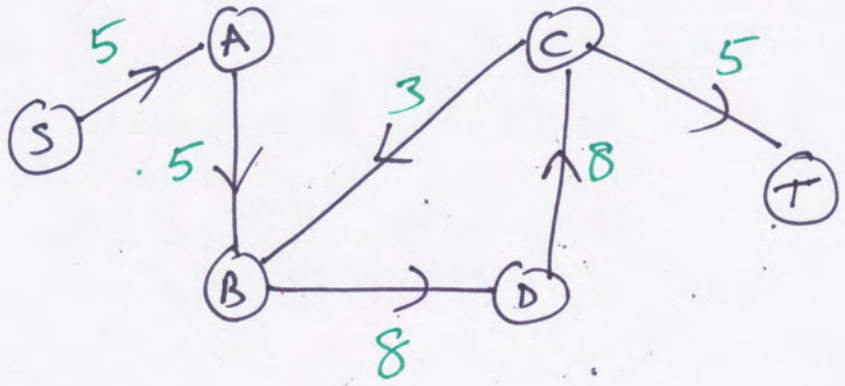


Edges with 0 flows remaining are removed.

Repeat the process

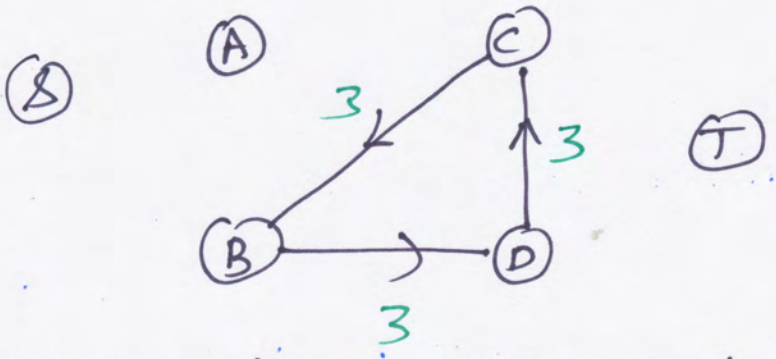
S-B-D-T: ~~30~~ 30

New remaining graph



S-A-B-D-C-T : 5

New remaining graph:



No S-t path

But cycles!

Decomposition: Path + cycle flows does not figure in F.

S-A-C-T : 15

S-B-D-T : 30

S-A-B-D-C-T : 5

F = 50

This decomposition takes polynomial time.

There may be many different path decompositions for a given "edge" flows that is feasible. There is always atleast one.

Hence the two formulations are both valid and we can transform from one to the other in poly. time.

Maximum Flow Algorithm (done in CS 6363)

1) Start with a feasible flow ; // an easily found one
when $0 \le f \le c$ is $f \equiv 0$
i.e $f(u,v) = 0 \ \forall (u,v) \in E$
$F = 0$

2) Given a feasible flow f,
Compute residual capacity $c^f(u,v) = c(u,v) - f(u,v) \ge 0$
$\forall u \in V, v \in V$
(including the case $(u,v) \notin E$ but $(v,u) \in E$) !!!

It is convenient here to think of $C : V \times V \to R_+$
$C(u,u) = \infty \ \forall u \in V$; $C(u,v)$ may not equal $C(v,u)$
and $f : V \times V \to R$, $f(u,v) = -f(v,u) \ \forall u \in V, v \in V$

---

(3) Construct a residual graph $G^f : [V; E^f]$

$$E^f : \{(u,v) : u \in V, v \in V; \ c^f(u,v) > 0\}.$$

(4) Ford-Fulkerson : Find a path from s to t in G^f

Edmonds-Karp : Find a path from s to t in G^f having minimum # of edges (Conveniently done using BFS)

(5) If no path exists (i.e BFS-tree starting at s does not include t)
STOP
F is maximum

(6) If $\exists P_{s,t}$ in G_f, change flows as follows

$$f^{new}(u,v) = \begin{cases} f^{old}(u,v) + C^f(P_{s,t}) & \text{if } (u,v) \in P_{s,t} \\ f^{old}(u,v) - C^f(P_{s,t}) & \text{if } (v,u) \in P_{s,t} \\ f^{old}(u,v) \end{cases}$$

Where $C^f(P_{s,t}) = \min_{(u,v) \in P_{s,t}} C^f(u,v) > 0$.

And go to Step 2 with new flows.

Remark: This algorithm maintains integrality throughout if $C(u,v)$ are integral and initial $f(u,v)$ are chosen integral (Can always be done).

This entire course is about integral solutions to problems when they exist.

Shown in CS6363

1. Algorithm maintains feasibility throughout
2. When it stops, we have maximum flow.
3. If $C(u,v)$ are integral, alg. stops in finite # of steps. [regardless of FF or EK]

An example to show FF version may not step if paths are not chosen wisely. [Taken from FF] Moreover, it does not converge to the correct value either for F. All this, of course, with irrational values for capacities.

Example: $G = [V; E]$

$V = [s, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, t]$

[10 nodes]

Edges: Four special edges

$A_1 : (x_1, y_1)$

$A_2 : (x_2, y_2)$

$A_3 : (x_3, y_3)$

$A_4 : (x_4, y_4)$

Additional edges: (s, x_i) $i = 1, 2, 3, 4$

(y_i, t) : $i = 1, 2, 3, 4$

(y_i, y_j) $i \neq j$; $i = 1 \cdots 4$, $j = 1 \cdots 4$

(x_i, y_j) $i \neq j$; $i = 1 \cdots 4$, $j = 1 \cdots 4$

(y_i, x_j) $i \neq j$; $i = 1 \cdots 4$, $j = 1 \cdots 4$

Capacities: all blue edge S (large number)

Capacities of red edges needs more explanation

Consider the recursion:

$$a_{n+2} = a_n - a_{n+1} \; ; \qquad n = 0, 1, \ldots$$

Solution: $a_n = r^n$; $r = \frac{-1+\sqrt{5}}{2} < 1$.

r is the positive root of

$$r^2 + r - 1 = 0$$

$$\sum_{n=0}^{\infty} a_n = S$$

Capacities of red edge:

$A_1 : a_0$
$A_2 : a_1$
$A_3 : a_2$
$A_4 : a_2$

At each step, some permutation of these four edges will have residual capacity of the form:

~~$a_n, a_{n+1}, a_{n+2}, a_{n+2}$~~ $(0, a_n, a_{n+1}, a_{n+1})$

As the algorithm progresses, after each 2 steps, n will increase by 1.

It is "clear" that the cut $[s, V-s]$ has capacity $4S$ (as does the cut $[V-t, t]$)

So maximum flow $\leq 4S$.

If is $4S$ as shown by

$s - x_1 - y_2 - t : S$

$s - x_2 - y_1 - t : S$

$s - x_3 - y_4 - t : S$

$s : x_4 - y_3 - t : S$

Now we will select (deliberately) a sequence of paths (infinite in number) with total flow approaching S. [Think as an adversary does]

Step 1: Use path $s - x_1 - y_1 - t : a_0$

Residual capacity of red edges

$A_1 : 0$ (in the direction $x_1 \to y_1$)

($\because$ $y_1 - x_1$: has a_0)

$A_2 : a_1$

$A_3 : a_2$

$A_4 : a_2$

Fits the pattern : $(0, a_n, a_{n+1}, a_{n+1})$; $n = 1$.

Step 2 & 3: (and general Step as well)

Suppose some permutation of red edges labeled A_1', A_2', A_3', A_4' have residual capacities (in the direction (x_i', y_i')): $(0, a_n, a_{n+1}, a_{n+1})$

We will use two paths and get to another permutation $(A_1'', A_2'', A_3'', A_4'')$ with residual capacities: $(0, a_{n+1}, a_{n+2}, a_{n+2})$. So this process goes on indefinitely. The two steps together increase F by a_n, giving a limiting

$$F = \sum_{n=0}^{\infty} a_n = S. < 4S.$$

Since all edges except A_1, A_2, A_3, A_4 have "large" capacities, they will not be the blocking edge in any path in the residual graph at any time. For the edges A_1, A_2, A_3, A_4 (or any permutation there of), we will mention the residual capacity in the direction (x_i, y_i). The residual capacity in the reverse direction is easily calculated.

Steps to go from $\{A_1', A_2', A_3', A_4'\}$ / $\{0, a_n, a_{n+1}, a_{n+1}\}$ $\rightarrow$ $\{A_1'', A_2'', A_3'', A_4''\}$ / $\{0, a_{n+1}, a_{n+2}, a_{n+2}\}$ with a total $\uparrow$ of a_n in F

Path #1: $s - \overbrace{x_2' - y_2'}^{A_2'} - \overbrace{x_3' - y_3'}^{A_3'} - t$: Increase in F a_{n+1}

a_n Large, a_{n+1} Large, Large

Since $a_n \downarrow n$, $a_{n+1} < a_n$. Hence bottleneck is A_3' with residual capacity a_{n+1}.

Residual Capacities AFTER flow $\uparrow$

$A_1' : 0$

$A_2' : a_n - a_{n+1} = a_{n+2}$ (Recall Recurrence)

$A_3' = 0$

$A_4' = a_{n+1}$

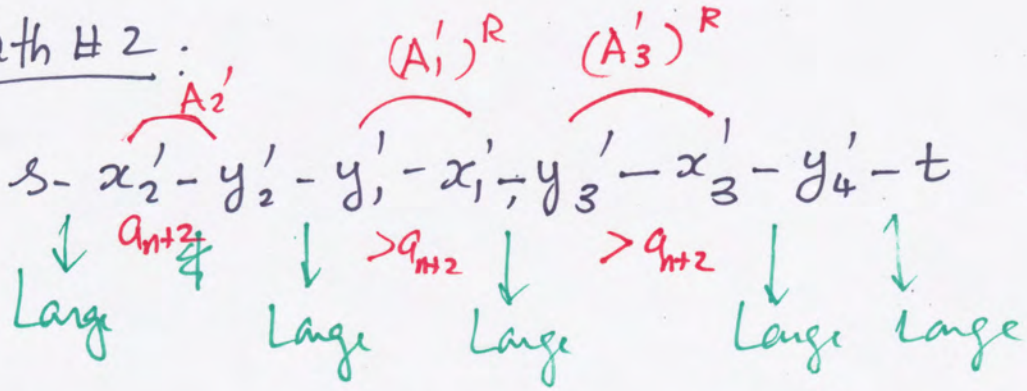
Reverse of $A_1' : y_1' \rightarrow x_1'$: Original capacity of (x_1', y_1') $\geq a_2 > a_n$ ~~$n \geq 2$~~

$\downarrow$ $A_2' : a_{n+2}$

$\downarrow$ A_3' : Original capacity of A_3 : $\geq a_2 > a_n$ $n \geq$

$A_4' : a_{n+1}$

Path # 2 :



Residual capacities after flow ↑ # 2

A_2' : ~~a_{n+2}~~ 0	=	A_1''	Pattern Maintained
A_1' : a_{n+2}	=	A_3''	
A_3' : a_{n+2}	=	A_4''	
A_4' : a_{n+1}	=	A_2''	

Total F ↑ in 2 Steps: $a_{n+1} + a_{n+2} = a_n$

Hence, this algorithm realization does NOT Stop in finite number of steps:

Total F = $\sum_{n=0}^{\infty} a_n = S < 4S$

So does NOT Converge to the Correct answer either.

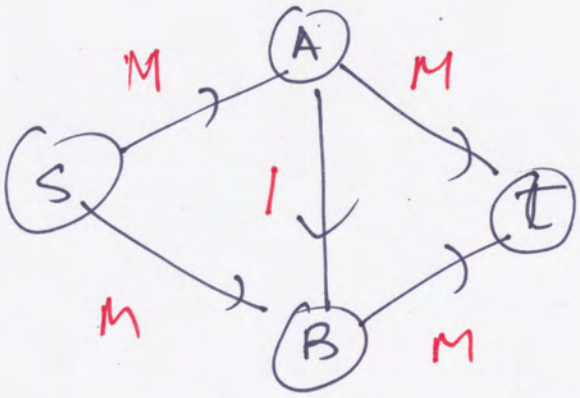
But capacities here are irrational numbers

An Even more powerful example due to J. Edmond

Example (J. Edmonds)

This prompts us to look for polynomial alg

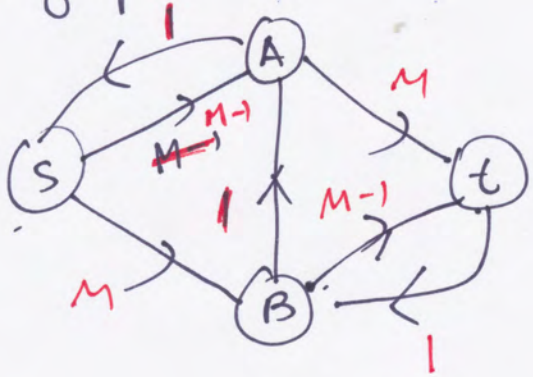
M: Large number ($\approx 10^6$)



Consider the following realization of F-F algorithm

Path #1 $S - A - B - t : 1$

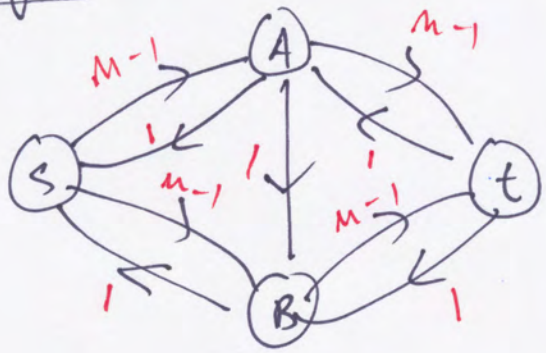
Residual graph.



Path #2 :

$S - B - A - t : 1$

Residual graph



This process is repeated M times for 2M augmentation

Certainly not polynomial in binary encription of data.

Finite Though!. Since C are integers