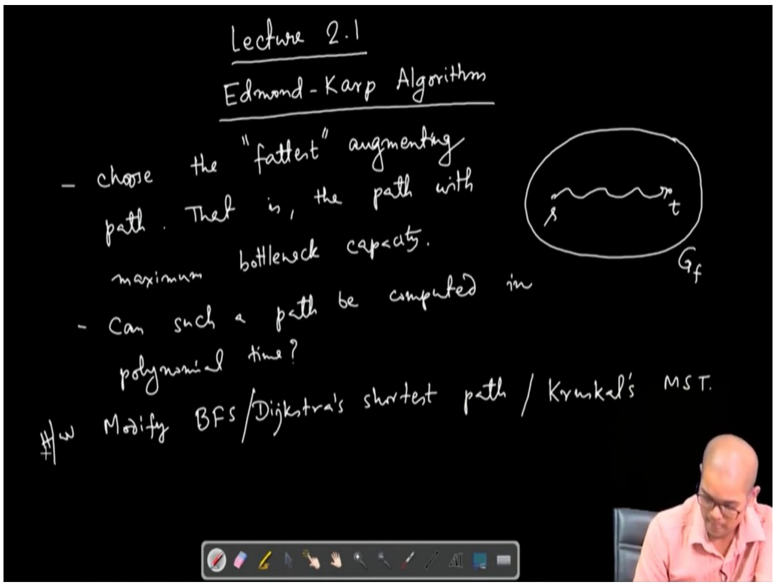
Selected Topics in Algorithm

Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module No # 02
Lecture No # 06
Maximum Bipartite Matching, Fattest Augmenting Path

Welcome in the last lecture we have seen how a flow can be decomposed into elementary flows path flows and cycle flows. Using this we will see algorithm again due to Edmonds and Karp which is a specialization of Ford-Fulkerson method.

(Refer Slide Time: 00:48)



Lecture 2.1

Edmond-Karp Algorithm

- choose the "fattest" augmenting path. That is, the path with maximum bottleneck capacity.
- Can such a path be computed in polynomial time?

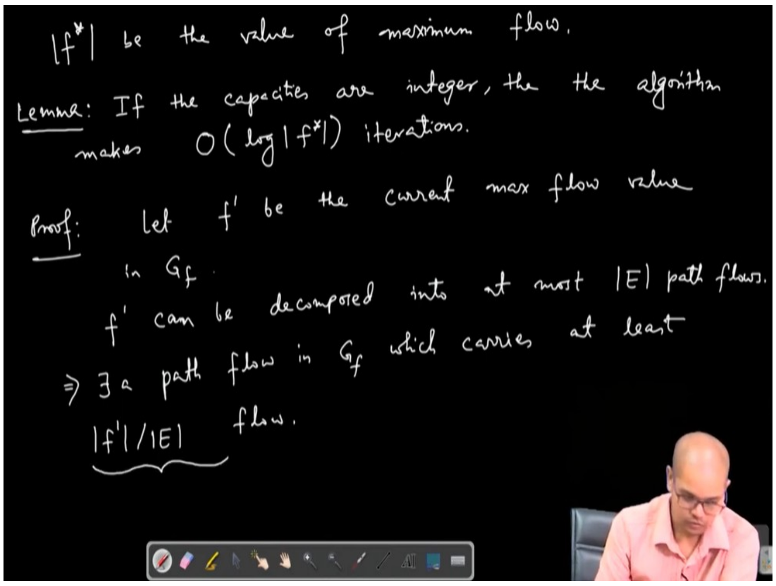
Hw Modify BFS/Dijkstra's shortest path / Kruskal's MST.

So this lecture 2.1 so let us recall earlier algorithm of Edmond Karp for computing maximum flow used to pick the shortest unawaited s to t path. But a more natural greedy approach would be to pick the path where maximum flow value can be pushed. So that particular approach is called choose that is the idea is to choose the fattest augmenting path that is the path with maximum bottleneck capacity.

So bottleneck capacity of a path is the minimum capacity of all the edges in the path you can push at max that much flow along the path. Now can such a path can be computed in polynomial time can search it path be computed in polynomial time? And the answer is yes I let you check you can either modify BFS or modify Dijkstra’s shortest algorithm or you modify appropriately the Kruskal’s minimum spending algorithm.

And I will let you check let you try this homework to get such a path. Now once I have such a path what I will do is exactly same it is same as Ford-Fulkerson methodology that you push that much flow maximum amount of flow along that path. And again the keep iterating unless there is no s to t path in the residual graph so how iterations are needed?

(Refer Slide Time: 04:54)



$|f^*|$ be the value of maximum flow.

Lemma: If the capacities are integer, the the algorithm makes $O(\log|f^*|)$ iterations.

Proof: let f' be the current max flow value in G_f.

f' can be decomposed into at most $|E|$ path flows.

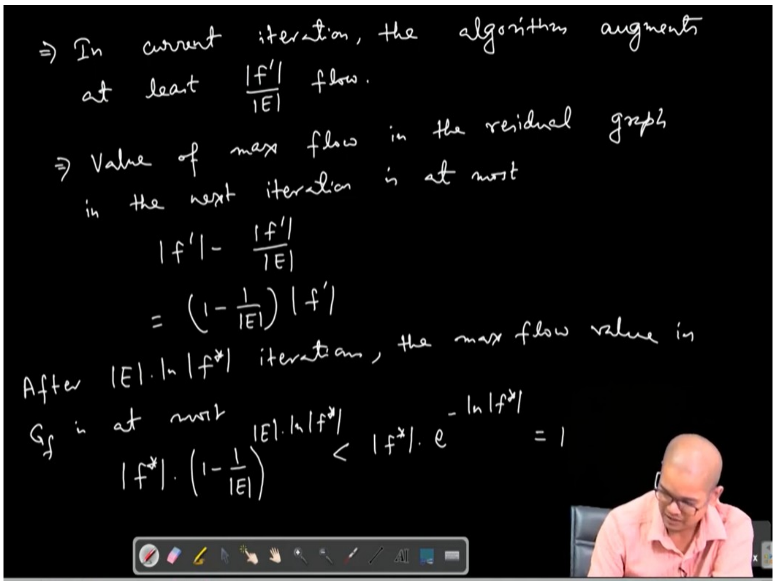
$\Rightarrow \exists$ a path flow in G_f which carries at least $\underbrace{|f'|/|E|}$ flow.

You know let f star be the value of maximum flow again if their capacities are irrational or arbitrary relational numbers then it is possible that it never converges it never holds it never terminates. It with more alteration it will go towards the optimal value but it never terminates that is possible. But we will show that you know if another Lemma if the capacities are integer then the algorithm makes $O(\log|f^*|)$ many iterations.

Proof so the basic idea is at every step we are augmenting substantial amount of flow and if we are doing this in every step then we should not take many iterations to reach the maximum value. So let f' be the current so considering any ith iteration and let f' be the current max flow value in the residual of G_f. So of course in the beginning G is G_f and f' is f^*.

Now we use the flow decomposition theorem that if prime can be decomposed into at most cardinality e many flow paths f' can be decomposed into at most cardinality e many path flows. Hence there exists path flow in G_f which carries at least value of f' by cardinality E flow. And this algorithm picks the path which were the maximum amount of flow would be pushed. So in; each iteration it must it in current iteration it augments at least this much flow.

(Refer Slide Time: 09:26)



⇒ In current iteration, the algorithm augments at least $\frac{|f'|}{|E|}$ flow.

⇒ Value of max flow in the residual graph in the next iteration is at most

$$|f'| - \frac{|f'|}{|E|}$$

$$= \left(1 - \frac{1}{|E|}\right)|f'|$$

After $|E| \cdot \ln|f^*|$ iterations, the max flow value in G_f is at most

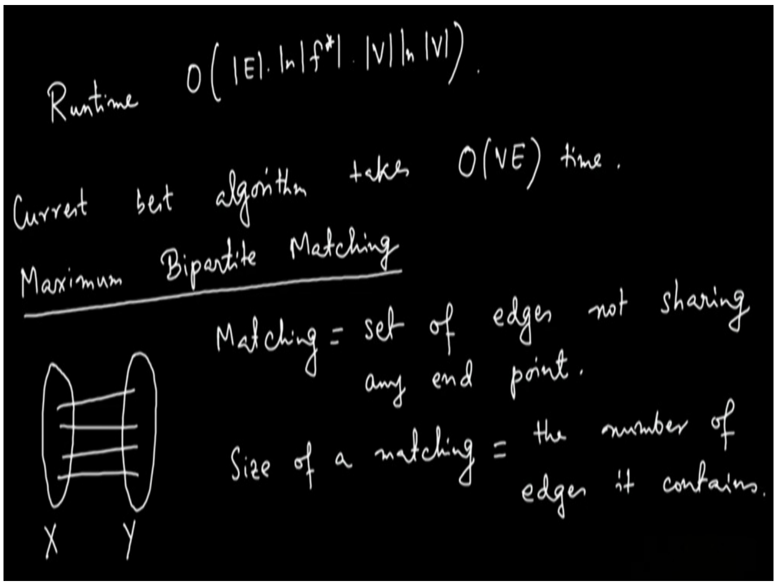
$$|f^*| \cdot \left(1 - \frac{1}{|E|}\right)^{|E| \cdot \ln|f^*|} < |f^*| \cdot e^{-\ln|f^*|} = 1$$

So in current iteration the algorithm augments at least value of $f^{'}$ by E. And so the value of maximum value iteration value of max flow in the residual graph in the next iteration in at most value of the $f^{'}$ is the next flow value in the current iteration and value of f prime by E was augmented so this is the thing that moves this which is 1 – 1 by E times. So the max flow value drops by a factor of 1 - 1 by E so hence after $|E| \times \ln(|f^*|)$ iterations the maximum value in G_f is at most.

So it is E times Lon extra iterations so it is if there should be a time there is a factor of E times foreign lon of, f star iterations. What should be the thing? At the beginning it is value of f star and this is 1 - 1 by E to be power $|E| \times \ln(|f^*|)$. So this is less than equal to $f^* e^{-\ln|f^*|}$ which is strictly less than 1 which is less than which is 1. This inequality is strict.

Now because so after $|E| \times \ln(|f^*|)$ menu iterations the number of the value of maximum value the value of maximum flow drops below one and because now the capacities of all the edges are integer the value of max flow after E times lon f terminal iterations must be 0. Hence this algorithm terminates after these many iterations. So what is the running time of this algorithm?

(Refer Slide Time: 14:00)



Runtime $O(|E| \cdot \ln|f^*| \cdot |V| \ln |V|)$.

Current best algorithm takes $O(VE)$ time.

Maximum Bipartite Matching

Matching = set of edges not sharing any end point.

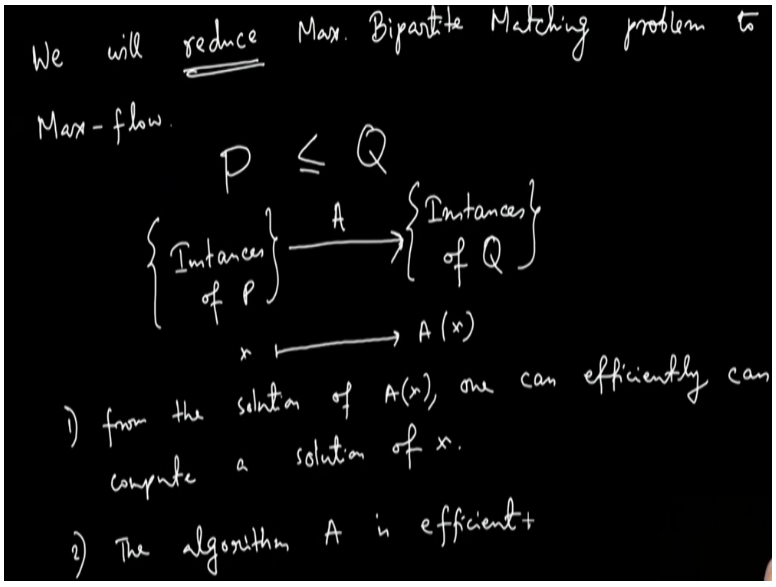
Size of a matching = the number of edges it contains.

Run time now we go of $|E| \times \ln(|f^*|)$ this is the number of iterations that it runs and in each iterations can be exit executed using Dijkstra's algorithm modified version of Dijkstra's algorithm which takes $V \log V$ time using Fibonacci series or E log time using standard min heap. So this is the running time and using that you know this if value f star is small compared to the number of edges then this algorithm is faster than the Edmond curve algorithm which picks the unawaited shortest s-t path for flow augmentation good.

So these are the 2 algorithm and the currently known based run time for computing max pro value current based algorithm current fastest algorithm takes $O(|V| \times |E|)$ time good. Now what I will see will show an important application there are many application of this flow many problems can be reduced into max pro and we will see once such application. So our application is maximum Bipartite matching.

So what is the Bipartite graph? It has 2 vertices can be partitioned into 2 sets X, Y X and Y they form induced independent sets there are no edges completely contained in X or completely contained in Y and all edges are cross edges only. And here I am looking for matching so matching is a set of edges containing edges not sharing any end point. And we are looking for maximum matching of a Bipartite graph so the size of the matching is the number of edges it contains.

(Refer Slide Time: 18:14)


We will reduce Max. Bipartite Matching problem to Max-flow.

$$P \leq Q$$

$$\begin{tikzcd}\{\text{Instances of } P\} \arrow[r, "A"] & \{\text{Instances of } Q\} \\ x \arrow[r, mapsto] & A(x)\end{tikzcd}$$

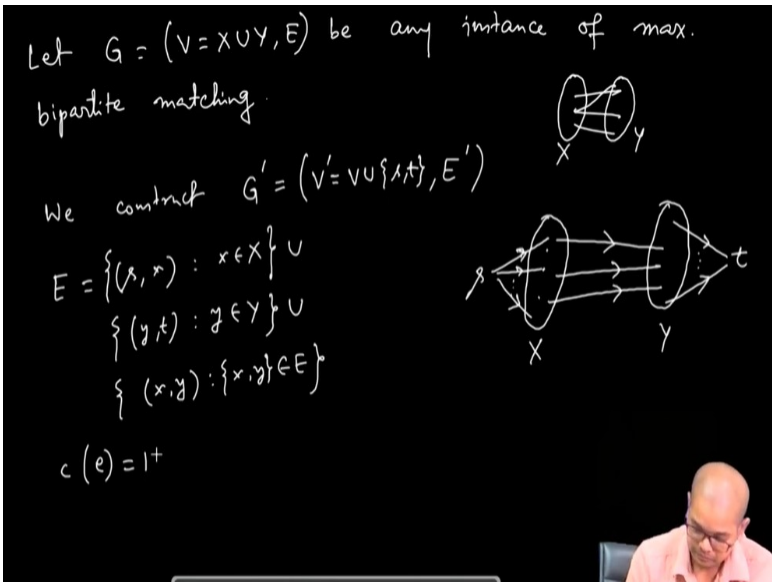
1) from the solution of $A(x)$, one can efficiently can compute a solution of x.

2) The algorithm A is efficient+

So what we will do is that we will reduce this maximum Bipartite matching problem to max flow. So and this reduction is a nice tool using which we can connect various problems now so let me define the slightly define this formally. Suppose I see that there is a reduction from problem P to problem Q it is basically an algorithm a which maps instances of p set of all instances of P it maps there is an algorithm A to instances of Q.

So reduction is basically an algorithm and so if this instance X is mapped to A of X we basically need 2 conditions one is that both are equivalent you know from the solution of A x one can efficiently that means in polynomial time can compute a solution of x that is it. And the second one is that algorithm A itself is efficient that means it runs in polynomial time. So here will reduce maximum Bipartite matching to max flow that means we will give an algorithm to construct instances of max flow from an instance of maximum Bipartite matching.

(Refer Slide Time: 21:25)


Let $G = (V = X \cup Y, E)$ be any instance of max. bipartite matching.

We construct $G' = (V' = V \cup \{s,t\}, E')$

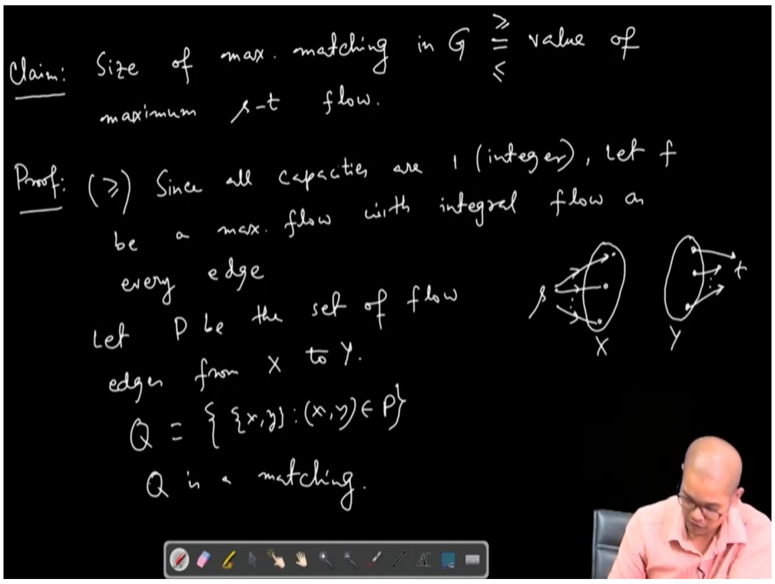
$E = \{(s,x) : x \in X\} \cup$
$\{(y,t) : y \in Y\} \cup$
$\{(x,y) : \{x,y\} \in E\}$

$c(e) = 1^{+}$

So let G = V = x union y, E so x and y are 2 part of the bipartite graph G be any instance of maximum bipartite matching. So we construct so I have a graph g x to y is x is y undirected graph there are various sort of edges. And we construct mu graph G' which is a flow graph where the vertices will be B prime is V union s and t is introduced to a new vertices is and t and I will describe the edges.

So basically I am taking this vertices only x and y and adding creating to new vertex is and t. And this E prime for each vertex in x I am adding in each from x to that vertex s to x for all x in x this union for all vertex in y I put an H from those vertex to T and all the edges from x to y which were undirected edges is direct them from x to y. Such that x, y this edge x, y belongs to E the matching instance. So these are the edges and what are the capacities we define the capacity of every edge to be 1.

(Refer Slide Time: 24:48)



Claim: Size of max. matching in G $\overset{\geq}{\underset{\leq}{=}}$ value of maximum s-t flow.

Proof: ($\geq$) Since all capacities are 1 (integer), let f be a max. flow with integral flow on every edge

Let P be the set of flow edges from X to Y.

$Q = \{\{x,y\} : (x,y) \in P\}$

Q is a matching.

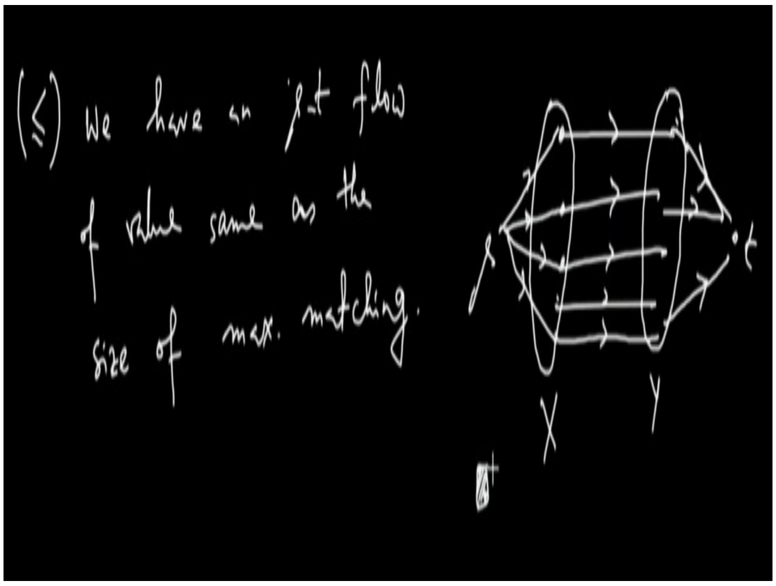
Now we claim that size of maximum matching in G is same as value of maximum s to t flow. So whenever we have quality one way to prove equality to show inequalities in both directions both greater than equal to and less than equal to. Proof so let us prove this direction that value of maximum matching in G is at least the value of the maximum s, t flow.

So let us take a maximum s, t flow and seems all capacities are 1 which will actually a integer we know that there is a maximum flow value where the flow values are integer the flow of each age is integer. So let f be max flow with integral flow on average but how does the capacities are only one. So intake and if the flow values are integral all the flow edges if the edge there is a flow the flow value will be 1.

Now you see that all the edges capacities are one so relate P be the set of this x, y set of flow edge which carries unit flow edges from x to, y. Let us define Q to be set of all edges x, y such that there is a flow edge from x to y. Now because the capacities of all vertex on x and all vertex on y are 1 the capacities of all incoming and outgoing the capacities of all incoming edges at x is 1 any 2 flow edge cannot be incident on 1 vertex of x.

Similarly if the only capacity of only outgoing edge of any vertex at y is y is 1 so the outflow of any vertex can be at most one in any vertex in y can be at most 1. So any 2 edges cannot have the same vertex in y common. So that means that y Q is a matching. So if I have a flow value is s, t flow then I have a matching with whose size at least the flow value.

(Refer Slide Time: 29:39)


(≤) We have an s-t flow of value same as the size of max. matching.

Similarly for the other direction I need to show that this is less than that if there is a matching. So let us draw this do this pictorially if there is a matching which matching means they do not share any end point. So these are the matching's then constructing s, t flow is very easy pick this end points of matching edges send one unit of flow along this to these vertices and send along one unit of flow along this matching it is from x to y and this matched vertex in .

You can send 1 unit of flow from that vertex to f so we have a flow and we have an s to t flow of value same as the size of maximum matching. And hence the value of maximum s t flow can only be more which concludes the proof. So it is as you can see you know from Kleinberg Carlos book provides lots of example and you will see that a various problems of various flavours they can reduced into this max flow problem. So we will stop here today thank you.