1

Introduction

- A little about me:
  - Carnegie Mellon University → Weizmann Institute
  - Hiking, guitar, volleyball.
- I love max-flows!

2

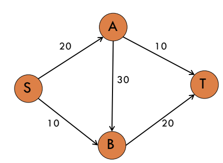
Learning Goals

- What is the max-flow problem?
- Why is it useful?
- What algorithm can we use to solve it?
- What's the connection to cuts?

3

Student networking

You decide to create your own campus network:
- You and three friends string together some network cables
- Because of capacity (due to cable type, distance, computer, etc) you can only send a certain amount of data to each person
- What is the maximum throughput you can you send from S to T?

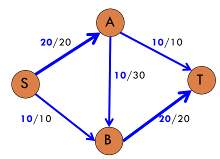


4

Student networking

You decide to create your own campus network:

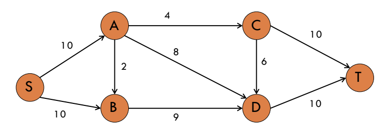
- You and three friends string together some network cables
- Because of capacity (due to cable type, distance, computer, etc) you can only send a certain amount of data to each person
- What is the maximum throughput you can you send from S to T?



30 units

5

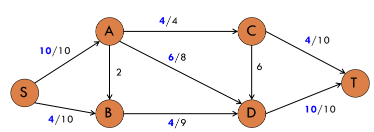
Another flow problem



How much water flow can we continually send from s to t?

6

Another flow problem



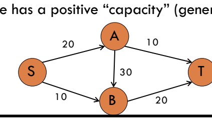
14 units

7

Flow graph/networks

Flow network

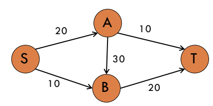
- Directed graph (V, E)
- Two distinguished vertices:
  - a source $s \in V$ with no incoming edges
  - a sink/target $t \in V$ with no outgoing edges
- Each edge has a positive "capacity" (generally, assume integers)



8

Flow constraints

What is a valid flow?



10

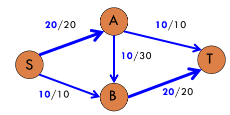
Flow constraints

What is a valid flow?

1. Flow along an edge cannot exceed its capacity

2. In-flow = out-flow for every vertex, except s, t

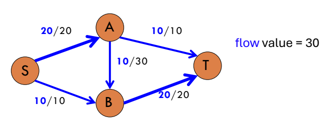
3. Flows are non-negative



11

Max flow problem

Given a flow network: *what is the maximum flow we can send from s to t that meets the flow constraints?*



12

Applications?

network flow
- water, electricity, sewage, cellular…
- traffic/transportation capacity

bipartite matching

sports elimination

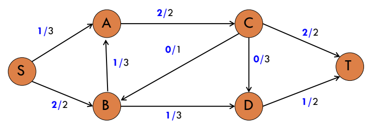
…

13

Check in

Pair up with a partner and think about:

- Is this a feasible flow?
- Is there a way to increase the flow?



15

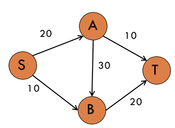
Learning Goals

- What is the max-flow problem?
- Why is it useful?
- What algorithm can we use to solve it?
- What's the connection to cuts?

16

Algorithmic idea

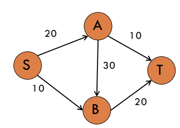
Keep sending flow along "augmenting paths"



18

Algorithmic idea

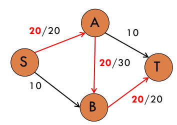
send flow on an augmenting path



19

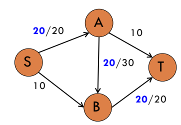
Algorithmic idea

send flow on an augmenting path



20

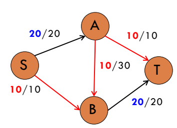
Algorithmic idea



Now what?

21

Algorithmic idea

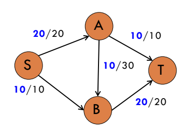


Total flow?

22

Algorithmic idea

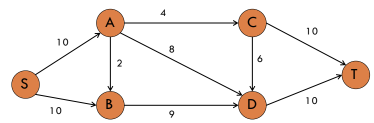
An augmenting path may need to undo some flow



30

23

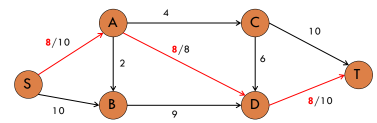
Algorithmic idea



24

Algorithmic idea

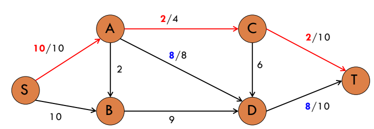
Find some augmenting path



25

Algorithmic idea

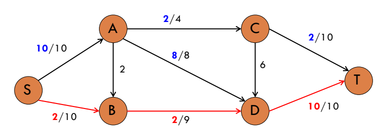
Find some augmenting path



26

Algorithmic idea

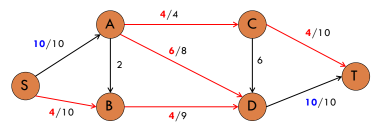
Find some augmenting path



27

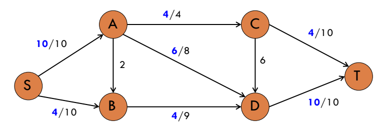
Algorithmic idea

Find some augmenting path



28

Algorithmic idea



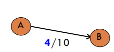
Are we done?

29

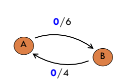
The residual graph

The *residual graph* G_f is constructed from G to help us understand if an augmenting path exists.

For each edge e in the original graph (G), we introduce two new edges.



becomes



G

G_f

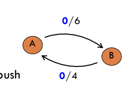
30

The residual graph

The *residual graph* G_f is constructed from G to help us understand if an augmenting path exists.

For each edge e in the original graph (G), we introduce two new edges:

- Forward edge:
  - has capacity = capacity(e)-flow(e)
  - this represents the remaining flow we can still push

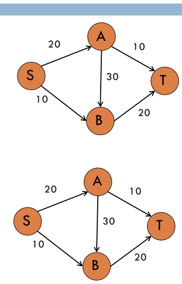


- Reverse edge:
  - put an edge in G_f in the *opposite direction* with capacity = flow(e)
  - this represents the flow that we can reverse / undo

31

Algorithm idea

G



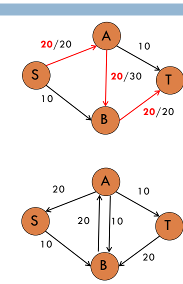
Gf

Find a path from *s* to *t* in Gf

32

Algorithm idea

G



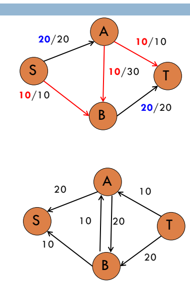
Gf

Find a path from *s* to *t* in Gf

33

Algorithm idea

G



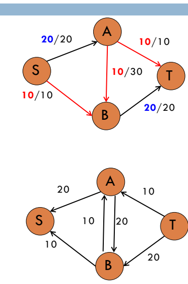
Gf

Find a path from *s* to *t* in Gf

34

Algorithm idea

G

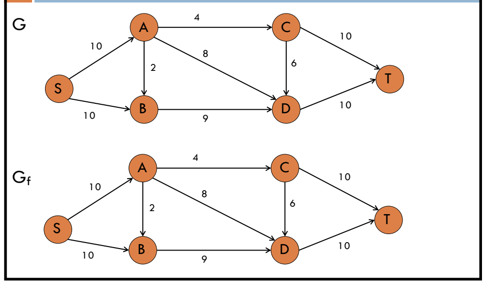


Gf

None exist… done!

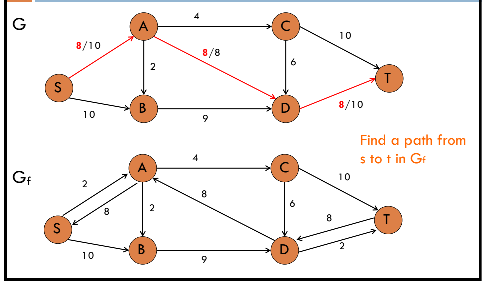
35

Algorithm idea



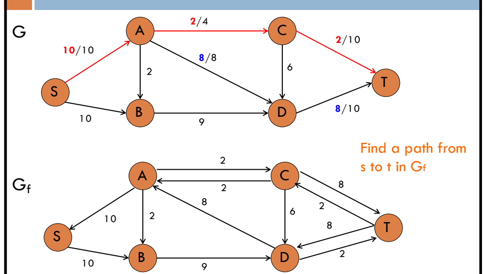
36

Algorithm idea



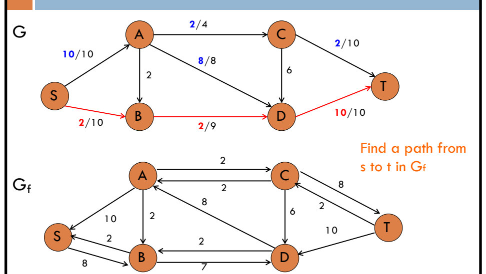
37

Algorithm idea



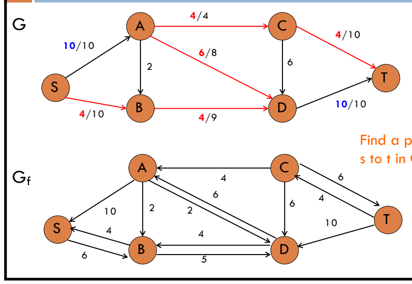
38

Algorithm idea



39

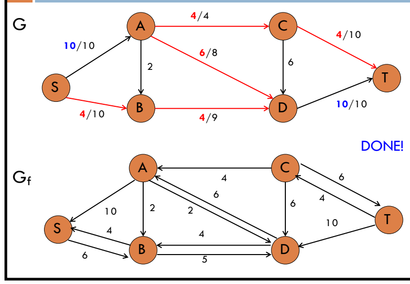
Algorithm idea



Find a path from s to t in G_f

40

Algorithm idea



DONE!

41

Ford-Fulkerson

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along the path as possible
    Gf = residualGraph(G)
  return flow
```

Questions?

42

Ford-Fulkerson: runtime?

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along path as possible
    Gf = residualGraph(G)
  return flow
```

43

Ford-Fulkerson: runtime?

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along path as possible
    Gf = residualGraph(G)
  return flow
```

- traverse the graph
- at most add 2 edges per original edge
- θ(V + E)

Can we simplify this expression?

44

Ford-Fulkerson: runtime?

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along path as possible
    Gf = residualGraph(G)
  return flow
```

- traverse the graph
- at most add 2 edges per original edge
- θ(V + E) = θ(E)

45

Ford-Fulkerson: runtime?

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along path as possible
    Gf = residualGraph(G)
  return flow
```

- BFS or DFS
- O(V + E) = O(E)

46

Ford-Fulkerson: runtime?

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along path as possible
    Gf = residualGraph(G)
  return flow
```

- max-flow!
- increases every iteration
- integer capacities, so integer increases

Can we bound the number of times the loop will execute?

47

Ford-Fulkerson: runtime?

```
Ford-Fulkerson(G, s, t)
  flow = 0 for all edges
  Gf = residualGraph(G)
  while a path exists from s to t in Gf
    send as much flow along path as possible
    Gf = residualGraph(G)
  return flow
```

- max-flow!
- increases every iteration
- integer capacities, so integer increases

Overall runtime? O(max-flow * E)

48

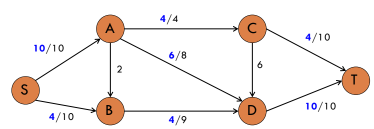
Learning Goals

- What is the max-flow problem?
- Why is it useful?
- What algorithm can we use to solve it?
  Ford-Fulkerson algorithm Runs in O(max-flow*E) time
- What's the connection to cuts?

49

Algorithm idea

Remember this graph?



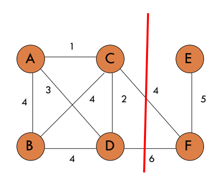
We know there is no augmenting path.

How do we know the flow is maximum? Enter cuts

50

Cuts

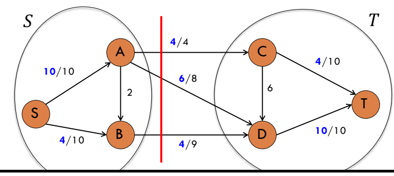
A cut is a partitioning of the vertices into two sets S and $T = V - S$



51

Flow across cuts

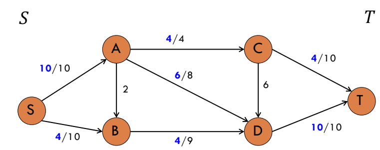
In flow graphs, we're interested in cuts that separate s from t, that is $s \in S$ and $t \in T$



52

Flow across cuts

In flow graphs, we're interested in cuts that separate s from t, that is $s \in S$ and $t \in T$

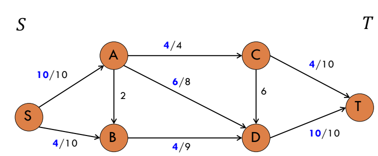


53

Flow across cuts

In flow graphs, we're interested in cuts that separate s from t, that is $s \in S$ and $t \in T$

Is this an $s, t -$ cut?

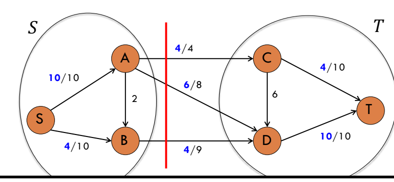


54

Flow across cuts

Some edges go "across" the cut, some stay within one side.

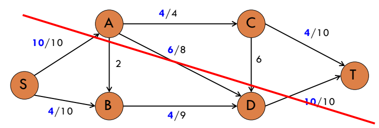
Two parameters of interest: the net flow and the capacity.



55

Flow across cuts

The net flow across a cut is the total flow from nodes in S to nodes in T ***minus*** the total from nodes in T to S

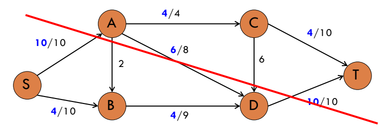


56

Flow across cuts

The net flow across a cut is the total flow from nodes in S to nodes in T ***minus*** the total from nodes in T to S

What is the net flow across this cut?

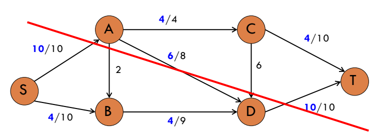


57

Flow across cuts

The net flow across a cut is the total flow from nodes in S to nodes in T ***minus*** the total from nodes in T to S

10+10-6 = 14

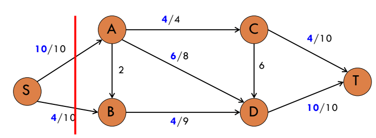


58

Flow across cuts

The net flow across a cut is the total flow from nodes in S to nodes in T ***minus*** the total from nodes in T to S

4+10 = 14

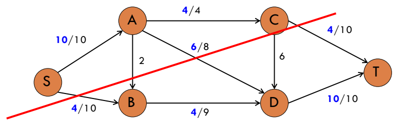


59

Flow across cuts

The net flow across a cut is the total flow from nodes in S to nodes in T *minus* the total from nodes in T to S

4+6+4 = 14

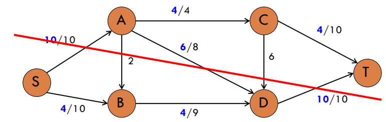


60

Flow across cuts

The net flow across a cut is the total flow from nodes in S to nodes in T *minus* the total from nodes in T to S

10+10-6 = 14

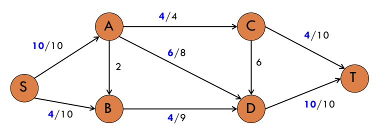


61

Flow across cuts

The net flow across ANY s,t-cut is the same and is the current flow in the network.

Why might this be?

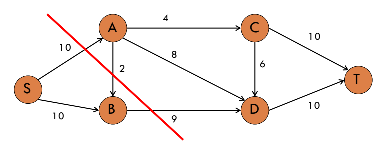


63

Capacity of a cut

The "**capacity of a cut**" is the maximum net flow that *could* be sent across it.

How do we calculate the capacity?

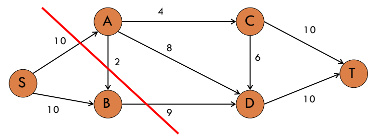


72

Capacity of a cut

The "**capacity of a cut**" is the maximum net flow that *could* be sent across it.

Capacity is the sum of capacities of edges from S to T



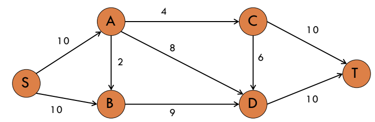
10 + 9 = 19

73

Capacity of a cut

The "**capacity of a cut**" is the maximum net flow that *could* be sent across it.

Capacity is the sum of capacities of edges from S to T

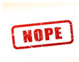


74

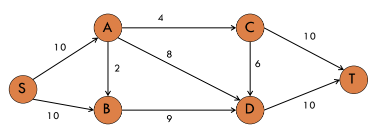
Bounding the flow

What if someone told you:

"The max flow in this network is 21"



"The max flow in this network is 20"

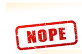


75

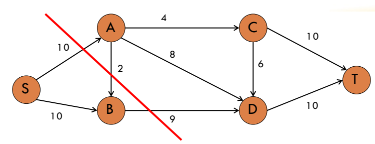
Bounding the flow

What if someone told you:

"The max flow in this network is 21"



"The max flow in this network is 20"

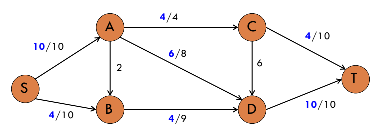


10 + 9 = 19

76

Maximum flow

For any cut where $s \in S$ and $t \in T$
- the net flow is at most the capacity
- the net flow across all cuts is the same

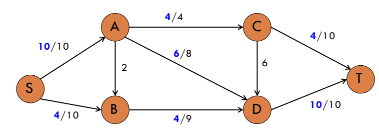


Is this the best we can do?

77

Maximum flow

For any cut where $s \in S$ and $t \in T$
- the net flow is at most the capacity
- the net flow across all cuts is the same



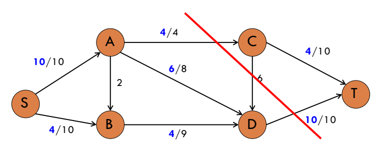
We can do no better than the minimum capacity cut!

78

Maximum flow

What is the minimum capacity cut for this graph?

Capacity = 10 + 4



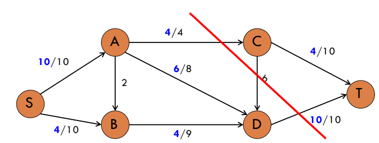
Is this the best we can do?

79

Maximum flow

What is the minimum capacity cut for this graph?

Capacity = 10 + 4



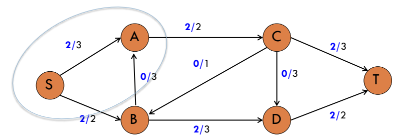
flow = minimum capacity, so we can do no better

80

Check in

The current flow in this network is 4.

- Can you find a cut with net flow equal to 4?
- Can you find a cut with capacity equal to 4?



81

Ford-Fulkerson: is it correct?

When the algorithm terminates, is it a maximum flow?

→ Termination condition:
- No s,t-path in the residual graph G_f

→ Look at the nodes S reachable from s in G_f by a path
- This is an s, t - cut!

→ This cut has net flow equal to its capacity

→ This is the maximum flow possible.

□

82

Learning Goals

What is the max-flow problem?

Why is it useful?

What algorithm can we use to solve it?

Ford-Fulkerson algorithm Runs in O(max-flow*E) time.

What's the connection to cuts?

In any flow network, the maximum st-flow is equal to the minimum capacity st-cut.

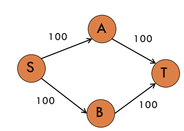
83

Runtime considerations

Ford –Fulkerson runs in time O(max-flow * E)

Can you construct a graph that *could get* this running time?

Hint:

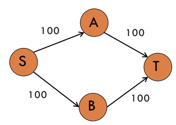


84

O(max-flow * E)

Can you construct a graph that *could get* this running time?

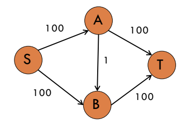
Hint:



85

O(max-flow * E)

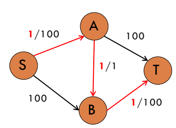
Can you construct a graph that *could get* this running time?



86

O(max-flow * E)

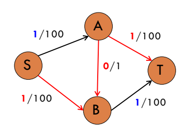
Can you construct a graph that *could get* this running time?



87

O(max-flow * E)

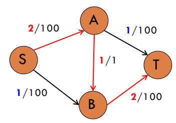
Can you construct a graph that *could get* this running time?



88

O(max-flow * E)

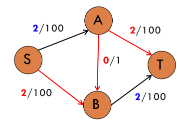
Can you construct a graph that *could get* this running time?



89

O(max-flow * E)

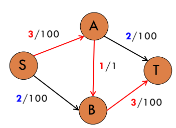
Can you construct a graph that *could get* this running time?



90

O(max-flow * E)

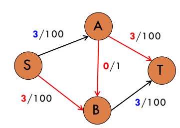
Can you construct a graph that *could get* this running time?



91

O(max-flow * E)

Can you construct a graph that *could get* this running time?



What is the problem here?
Could we do better?

92

Faster variants

Edmonds-Karp
- Select the *shortest path* (in number of edges) from s to t in G_f
  - How can we do this?
    - use BFS for search
- Running time: $O(V E^2)$
  - avoids issues like the one we just saw
  - see the book for the proof
  - or http://www.cs.cornell.edu/courses/CS4820/2011sp/handouts/edmondskarp.pdf

preflow-push (aka push-relabel) algorithms
- $O(V^3)$

93

Other variations…

Method	Complexity
Linear programming	
Ford–Fulkerson algorithm	$O(E \max\|f\|)$
Edmonds–Karp algorithm	$O(VE^2)$
Dinitz blocking flow algorithm	$O(V^2E)$
General push-relabel maximum flow algorithm	$O(V^2E)$
Push-relabel algorithm with *FIFO* vertex selection rule	$O(V^3)$
Dinitz blocking flow algorithm with dynamic trees	$O(VE \log(V))$
Push-relabel algorithm with dynamic trees	$O(VE \log(V^2/E))$
Binary blocking flow algorithm [1]	$O(E \min(V^{2/3}, \sqrt{E}) \log(V^2/E) \log U)$
MPM (Malhotra, Pramodh-Kumar and Maheshwari) algorithm	$O(V^3)$

http://en.wikipedia.org/wiki/Maximum_flow

TABLE I. POLYNOMIAL-TIME ALGORITHMS FOR THE MAXIMUM FLOW PROBLEM[a]

Algorithm no.	Date	Discoverer	Running time	References
1	1969	Edmonds and Karp	$O(nm^2)$	[5]
2	1970	Dinic	$O(n^2m)$	[4]
3	1974	Karzanov	$O(n^3)$	[18]
4	1977	Cherkasky	$O(n^2m^{1/2})$	[3]
5	1978	Malhotra, Pramodh Kumar, and Maheshwari	$O(n^3)$	[21]
6	1978	Galil	$O(n^{5/3}m^{2/3})$	[11]
7	1978	Galil and Naamad; Shiloach	$O(nm(\log n)^2)$	[12, 25]
8	1980	Sleator and Tarjan	$O(nm \log n)$	[27, 28]
9	1982	Shiloach and Vishkin	$O(n^3)$	[26]
10	1983	Gabow	$O(nm \log U)$	[10]
11	1984	Tarjan	$O(n^3)$	[31]
12	1985	Goldberg	$O(n^3)$	[14]
13	1986	Goldberg and Tarjan	$O(nm \log(n^2/m))$	[16, 15]
14	1986	Ahuja and Orlin	$O(nm + n^2 \log U)$	[1]

[a] Algorithm 13 is presented in this paper.

http://akira.ruc.dk/~keld/teaching/algoritmedesign_f03/Artikler/08/Goldberg88.pdf

94

Network flow properties

If one of these is true then all are true (i.e. each implies the the others):

f is a maximum flow

G_f (residual graph) has no paths from s to t

$|f|$ = minimum capacity cut

95

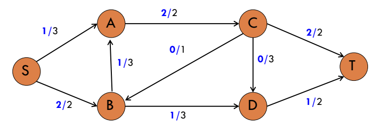
Handout

97

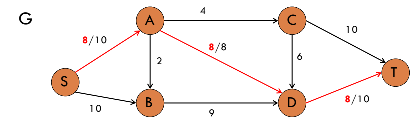
Check in

Pair up with a partner and think about:

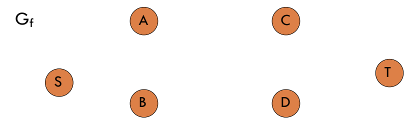
- Is this a feasible flow?
- Is it maximum?



98



Calculate residual graph

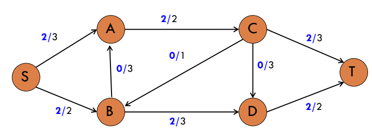


99

Check in

The current flow in this network is 4.

- Can you find a cut with net flow equal to 4?
- Can you find a cut with capacity equal to 4?



100