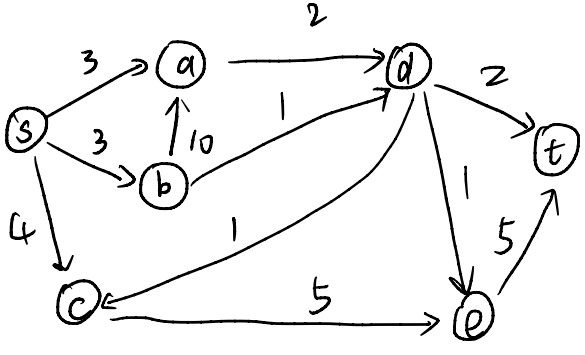
1. Shortest Paths on Graphs using Bellman-Ford Algorithm.

(a) The Bellman-Ford execution is shown in the following table.

	0	1	2	3	4
t	t/0	t/0	t/0	t/0	t/0
s	∞	∞	∞	b/6	b/6
a	∞	∞	d/4	d/4	d/4
b	∞	∞	d/3	d/3	d/3
c	∞	∞	e/10	e/10	e/10
d	∞	t/2	t/2	t/2	t/2
e	∞	t/5	t/5	t/5	t/5

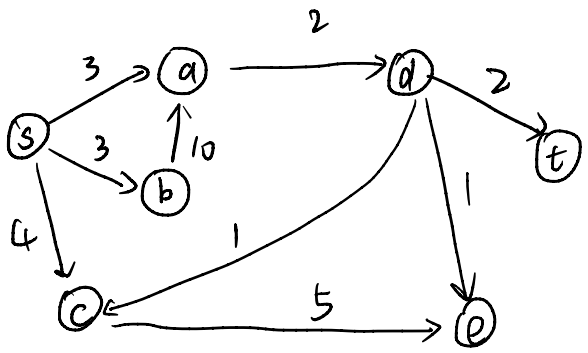


node	next hop	Distance
s	b	6
a	d	4
b	d	3
c	e	10
d	t	2
e	t	5

we can read from the column of 3 or 4.

	dis	route
t→t	0	t→t
s→t	6	s→b→d→t
a→t	4	a→a→t
b→t	3	b→d→t
c→t	10	c→e→t
d→t	2	d→t
e→t	5	e→t

(b)



Does the algorithm converge?

NO.

The intermediate steps are as follows.

↓ e detects the change and switches from next hop t to null with cost of ∞.

c detects the change on e and switch the cost with ∞.

(2)

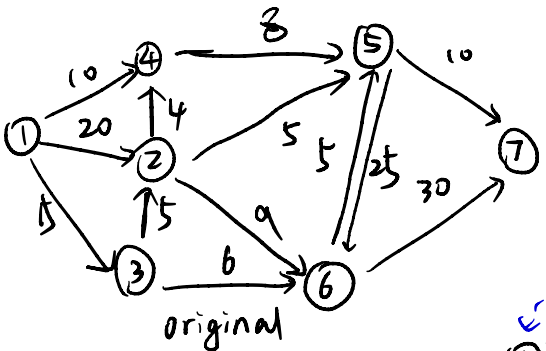
b switches next hop d to a with cost of 14.

S switches next hop b to a with cost of 7

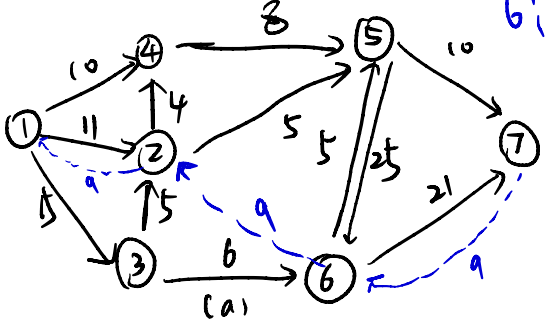
Node	Next hop	Distance
s	a	7
a	d	4
b	a	14
c	e	∞
d	t	2
e	null	∞

2. Finding Max Flow.

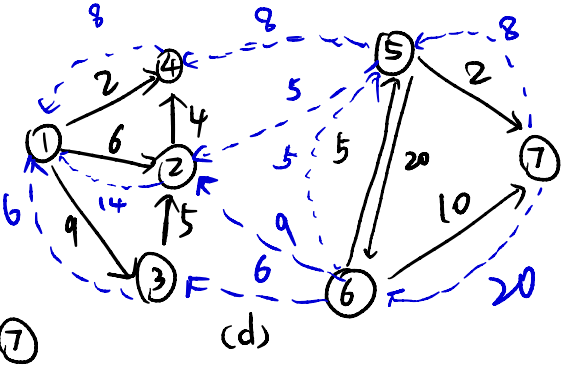
(a)



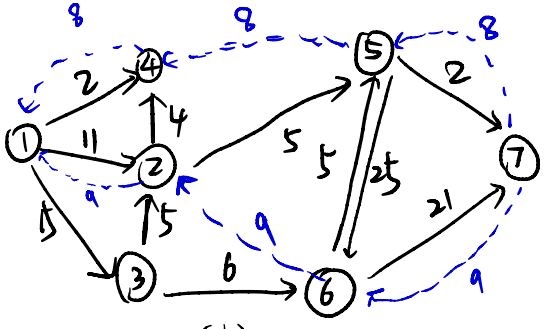
original



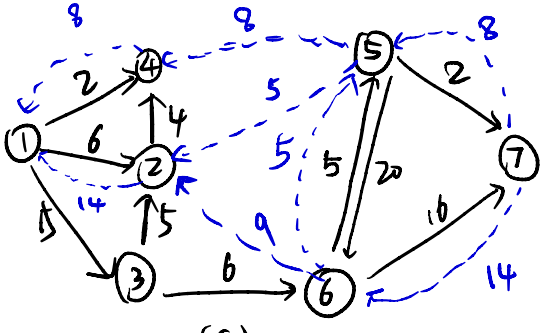
(a)



(d)



(b)



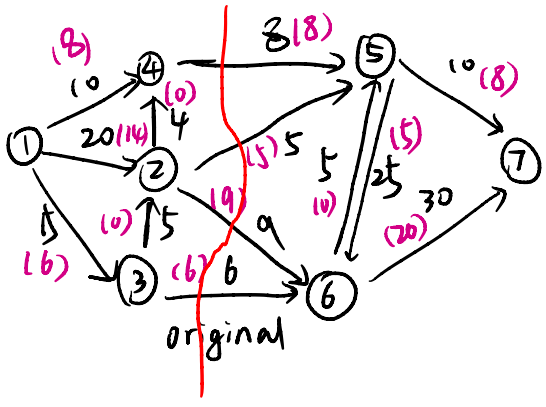
(c)

max-flow is

$(10+20+5)-(2+6+9)$

$= 45-17 = 28$

(b)



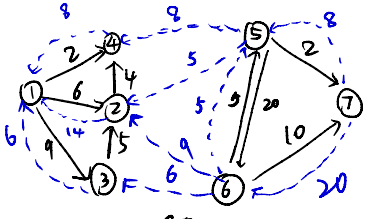
$8+5+9+6 = 28 = v(f)$

As the figure shows, the min-cut consists of edges $\{4\to5, 2\to5, 2\to6, 3\to6\}$ and has capacity 28. (equal to the max flow value)

What's more, the min-cut can be found by running BFS.

(c)

i. Algorithm.



- Reduce the flow on edge 2–6 by $\Delta C = 9-2=7$
- Find an augmenting path P_{21} on GF, from head to the source. Send flow $\max\{flow(P_{21}), \Delta C\}$. Repeat finding augmenting paths and pushing back flow until the total decrease is ΔC.
- Find an augmenting path P_{76} on GF, from sink to tail. Send flow $\max\{flow(P_{76}), \Delta C\}$. Repeat finding augmenting paths and pushing back flow until the total decrease is ΔC.

Because the edge was part of the min-cut, we can stop. otherwise, we need run the algorithm until we saturate it.

ii The new flow has value $\Delta C = 28 - (9-2) = 21$, this is also the new capacity of the cut $\{4\to5,\ 2\to5,\ 2\to6,\ 3\to6\}$ and therefore is the new max flow.

iii

(1) we need run BFS on the residual graph twice to finding augmenting paths from the ends of the affected edge to the source and sink. $O(m+n)$

(2) Augmenting each path $O(n)$

(3) Updating G_F takes $O(m)$

we need at most ΔC such iterations and $n \le 2m$. ($\Delta C = 9-2=7$)

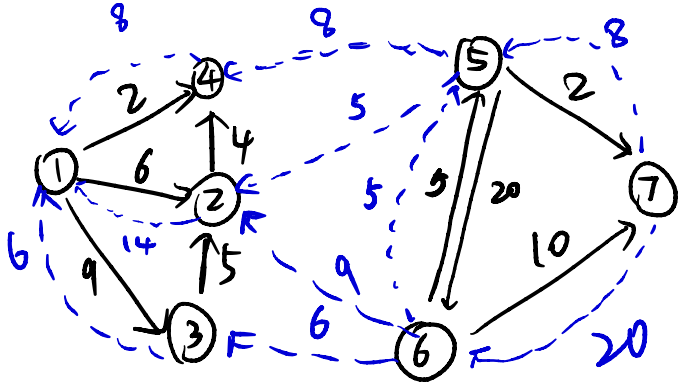
so running time is $O(\Delta C \cdot m)$.

iv

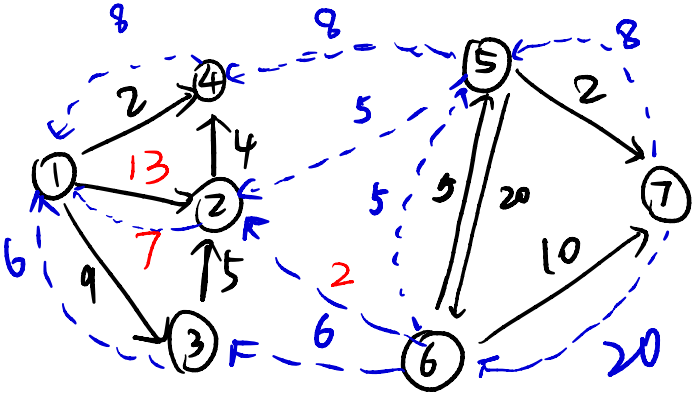
$\Delta C = 9-2=7$ and the new flow is the same as before, with the changes

$f(1\to2) = 14-7=7$ $f(2\to6) = 9-7=2$ $f(6\to7) = 20-7=13$.

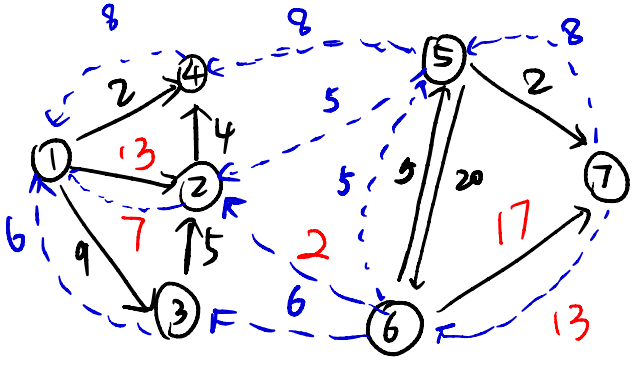
The new max flow is value $28-7=21$



GF



GF



The flow value =

$(10+20+15)-(2+13+9)=21$

3.

(a) The flow network is built as follow. The node v_i for each client i and the node w_j for the specific base station j. If a client can be connected to a base station within the required distance, we add an edge (v_i, w_j) of capacity 1. we use a super-source s to connect every client with the capacity of 1. And the we connect each of the base station nodes to a super-sink t by an edge of capacity L.

(b) If and only if there is a flow of value n from node s to node t. we say that there is a feasible way to connect all clients to base stations.

feasible solution $\Rightarrow$ the value of flow is n

on the feasible connection, we can send one unit of flow from s to t along the path $s \to v_i \to w_j \to t$, where v_i is connected to w_j. And this does n't violate the capacity condition, although the limits on edge (w_j, t).

conversely.

each client has a path $s \to c_i \to s_j \to t$ and the capacity constrain on edge $s_j \to t$ will be valid.

Therefore, each client is connected with a basestation and no more than L clients occupy one base station.

(c) The running time equals to solve a max-flow problem on a graph with $O(n+k)$ nodes and $O(nk)$ edges. The total complexity is $O(n^2k)$.