1.

① Choose the job that starts last.

we can select the activities in reverse order.
that is first select the last activity. ~~...~~

- we can select the ~~last~~ activity in every step by selecting the last activity to start that is compatible with all previously selected activities.

Proof:

Suppose A is our solution and O is optimal solution.

$$A = (i_k, \dots i_3, i_2, i_1)$$

$$O = (j_m, \dots j_3, j_2, j_1)$$

Since O is optimal set of activities

$$k \leq m$$

$\Rightarrow \forall\ i_r \in A,\ j_r \in O$

compare start time

$S_{i_r} \geq S_{j_r}$ $\Rightarrow$ Because in our algorithm we pick activity that starts late.

⇒ suppose if A is not optimal Solution,

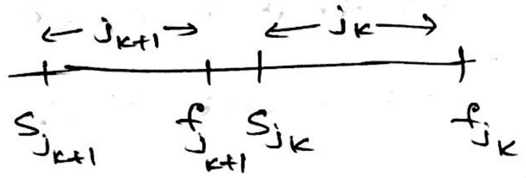
then
$k < m$

∴ there exist j_{k+1} activity which is in O

$j_{k+1} \in O$

j_k and j_{k+1} should be compatible.

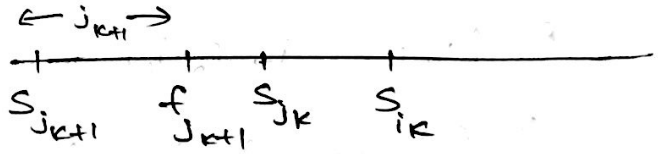
$s_{j_k} > s_{j_{k+1}}$



we already have

$s_{i_k} \geq s_{j_k}$

∴



∴ $s_{i_k} > s_{j_{k+1}}$

since $s_{j_{k+1}}$ will finish before s_{j_k}, it will definetly finish before s_{i_k}.

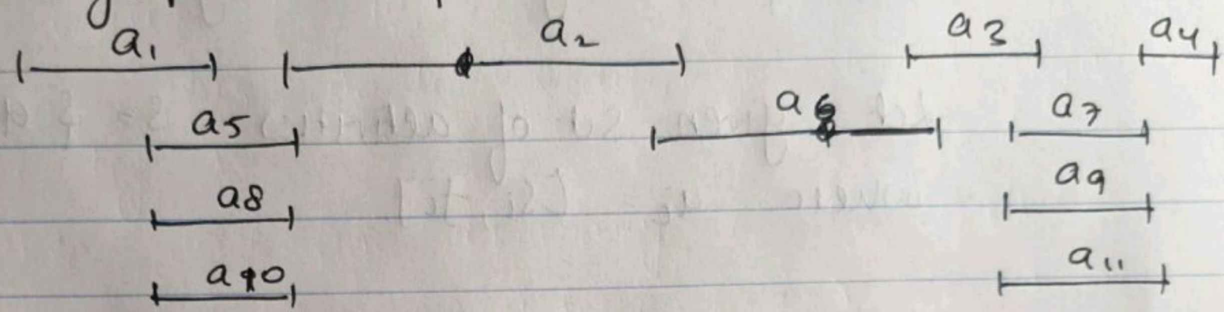
∴ j_{k+1} and i_k activities are compatible.

∴ After i_k, our algorithm will pick j_{k+1}.

∴ our solution is optimal Solution.

b) Choose the job that conflicts with fewest ~~ob~~ other jobs:

This is not a greedy approach., consider. below graph to prove.

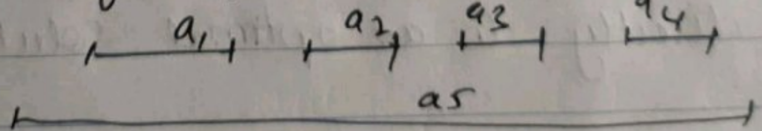


using this approach, the solution will. make us to choose first activity a_6 and than ~~ag~~ activity a_4

while the most appropriate solution will be (a_1, a_2, a_3, a_4).

(c) Choose. the job of longest duration

This is not a greedy approach, consider below graph for prove



Using this approach solution will be choosing only activity a_5, while the most appropriate solution will be $\{a_1, a_2, a_3, a_4\}$.

Scanned by CamScanner

Scanned by CamScanner

Problem 2.

(1) We have $c_1 = 1$, $c_2 = 5$, $c_3 = 10$, $c_4 = 25$, $c_5 = 50$ and $c_6 = 100$. Does the greedy algorithm always give the change with the fewest coins? Either prove it does, or provide a couterexample.

Solution. - Use the greatest value coins for the existing amount and to use as many of those coins possible without exceeding the existing amount.

For example: if $d = 150$.
then, the largest value coin available $= 50$.
hence. we use 3×50 coins. $= 150$.

This is a greedy aproach. since, we chose the best solution for current step without regard of optimal solution.

Proof: (1) Lets consider coins with values $c_1 = 1$ and $c_2 = 5$.
We can use atmost 4 coins with value $c_1 = 1$ because any number larger than or equal to 5 will be replaced with coin $c_2 = 5$.
Choosing $c_2 = 5$ would reduce total coins by 4.

Scanned by CamScanner

Scanned by CamScanner

(2) Lets consider coins with values $c_1 = 1$, $c_2 = 5$ & $c_3 = 10$.

We can use only 1 coin with value $c_2 = 5$ atmost, since any number greater than or equal to 10 will be replaced with coin value $c_3 = 10$.

This would reduce the total coins number by 1.

We will use as many $c_3 = 10$ coins as possible before considering $c_2 = 5$ & $c_1 = 1$ coins.

(3) Similarly we would consider choosing the coins of different values depending on the number.

$c_1 = 1$, atmost 4 coins

$c_2 = 5$, atmost 1 coin.

$c_3 = 10$, atmost 2 coins, any number or remainder greater than or equal to 30 will replaced by $c_4 = 25$ coin

$c_4 = 25$, atmost 1 coin, any number or remainder greater or equal to 50 will be replaced by $c_5 = 50$ coins

$c_5 = 50$, atmost 1 coin, any number or remainder greater of equal to 100 will be replaced by $c_6 = 100$ coin.

Scanned by CamScanner

for example $d = 149$.

① The greatest value coin $C_6 = 100$

~~hen~~ remainder = $149 - 100 = 49$.

② Then greatest value coin $C_4 = 25$.

remainder $= (149 - 100) - 25 = 24$

③ Then greatest value coin $C_3 = 10$

remainder $= (24 - 10) = 14$

④ Then greatest value coin $C_3 = 10$

remainder $= 14 - 10 = 4$

⑤ Then greatest value coin $C_1 = 1$

$4 - (1 \times 4) = 0$.

Final solution $(1 \times 100) + (1 \times 25) + (2 \times 10) + (4 \times 1)$

$= 100 + 25 + 20 + 4$

$= 149$.

Total coins = 8. coins.

Scanned by CamScanner

Scanned by CamScanner

Problem 2

(e) Suppose the denominations are consecutive powers of an integer $b \geq 2$; that is $c_1 = 1$, $c_2 = b$, $c_3 = b^2$, and so on. Does the greedy algorithm always give the change with fewest coins? Either prove that it does or provide a counter example.

Soln:

① Reasoning is very similar to previous question. Lets consider first two types of coins i.e. $c_1 = 1$ & $c_2 = b$.

Atmost we can $b-1$ $c_1 = 1$ value coins because any number larger than or equal to b will be replaced by $c_2 = b$ value coin.

Hence any remainder greater than b, will allow to use as ~~many~~ coin will $c_1 = 1$ and $c_2 = b$ and we will use as many coins of $c_2 = b$ as possible before considering $c_1 = 1$.

② Lets consider coins with values c^{n-1} and c^n. At most we can use $(c-1) \times c^{n-1}$ coins. Any number larger than c^n or equal to c^n will be replaced by c^n coin. This will reduce total numbers of coins by $c-1$.

Scanned by CamScanner

For example.

$b = 2.$ $\quad C_2 = 2^1 = 2$

coins $\quad C_1 = 1$, $C_3 = 2^2 = 4$, $C_4 = 2^3 = 8.$

$d = 15.$

(1) We will choose coin with value $C_4 = 8.$

remainder: $15 - 8 = 7.$

(2) we will choose coin with value $C_3 = 4$

remainder $= 7 - 4 = 3.$

(3) we will choose coin with value $C_2 = 2$

remainder $= 3 - 2 = 1$

(4) we will choose coin with value $C_1 = 1$

remainder $= 1 - 1 = 0$

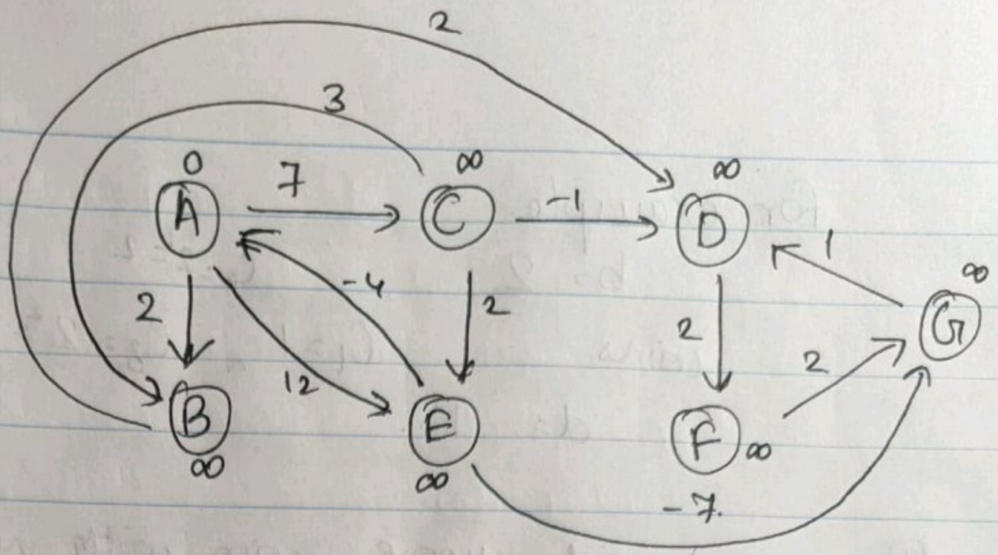
Total coins $= (1 \times 8 \text{ coin}) + (1 \times 4 \text{ coin}) + (1 \times 2 \text{ coin}) + (1 \times 1 \text{ coin})$

$= 4$ coins.

Scanned by CamScanner

Problem 3.

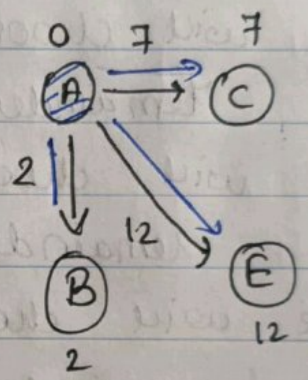
① Sol^n:-



Step 1.

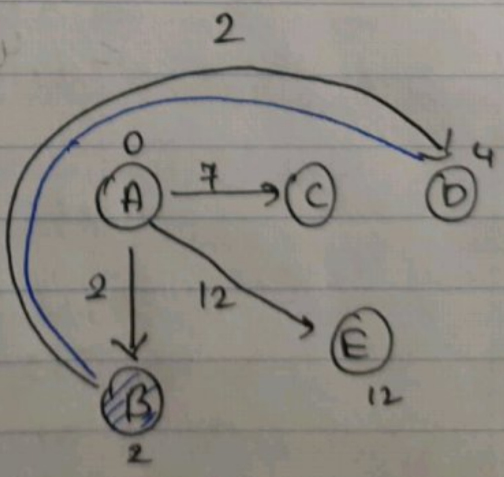
(vertex choosen = A)

vertex	known	Distance	Path
A	✓	0	A
B		2	A → B
C		7	A → C
D		∞	
E		~~∞~~ 12	A → E.
F		∞	
G		∞	



Step 2. (vertex choosen = B)

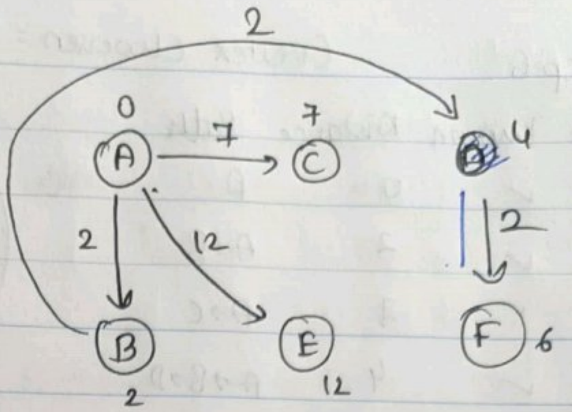
vertex	known	Distance	Path
A	✓	0	A
B	✓	2	A → B
C		7	A → C
D		4	A → B → D
E		12	A → E
F		∞	
G		∞	



Scanned by CamScanner

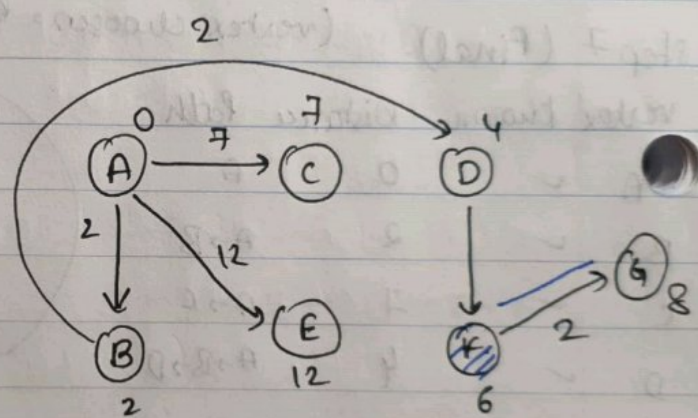
Step 3. (vertex Choosen = D.)

vertex	known	Distance	Path
A	✓	0	A
B	✓	2	A→B
C		7	A→C
D	✓	4	A→B→D
E		12	A→E
F		6	A→B→D→F
G		∞	



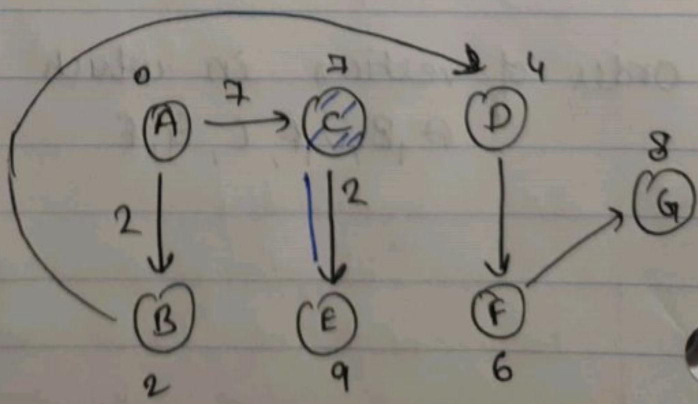
Step 4 (vertex choosen = F)

vertex	known	Distance	Path.
A	✓	0	A
B	✓	2	A→B
C		7	A→C
D	✓	4	A→B→D
E		12	A→E
F	✓	6	A→B→D→F
G		8	AB→D→F→G



Step 5 (vertex choosen = C)

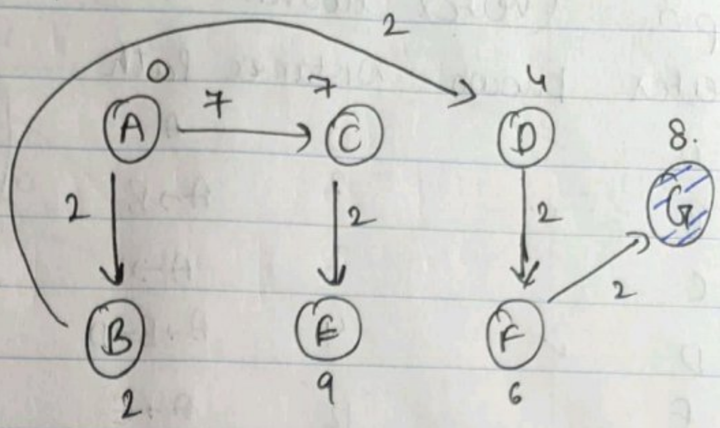
vertex	Known	Distance	Path
A	✓	0	A
B	✓	2	A→B
C	✓	7	A→C
D	✓	4	A→B→D
E		9	A→C→E
F	✓	6	A→B→D→F
G		8	A→B→D→F→G



Scanned by CamScanner

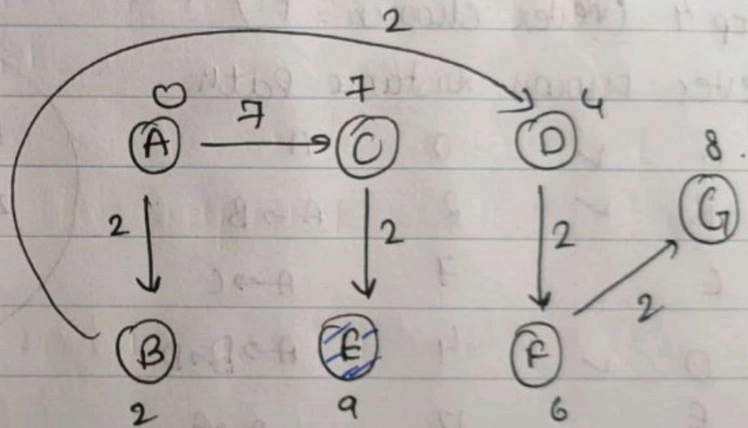
step 6 (vertex choosen = G)

vertex	known	Distance	Path
A	✓	0	A
B	✓	2	A→B
C	✓	7	A→C
D	✓	4	A→B→D
E		9	A→C→E
F	✓	6	A→B→D→F
G	✓	8	A→B→D→F→G



step 7 (Final) (vertex choosen = E)

vertex	known	Distance	Path
A	✓	0	A
B	✓	2	A→B
C	✓	7	A→C
D	✓	4	A→B→D
E	✓	9	A→C→E
F	✓	6	A→B→D→F
G	✓	8.	A→B→D→F→G



order of vertices in which they are marked as known
A, B, D, F, C, G, E

Scanned by CamScanner

Problem 3

2) Soln.

vertices for which the wrong paths are computed.

(i) vertex G.

computed Path = A → B → D → F → G = 8

Correct Path = A → C → E → G = 2

(ii) vertex D:

computed Path = A → B → D ~~A → C → E → G → D~~ = 4

Correct Path = A → C → E → G → D = 3

(iii) vertex F

computed Path : A → B → D → F = 6

Correct Path : A → C → E → G → D → F = 5.

Scanned by CamScanner

Scanned by CamScanner

4. Suppose maximum flow of network is f.

let (u,v) be any pair of vertices.

Assume that ~~$0<f(u,v) \le f(v,u)$~~

$$0 < f(u,v) \le f(v,u)$$

~~By~~ If both the incoming and outgoing flows of any pair of vertices (suppose (u,v)) are decreased by the same amount, the net flow between the vertices will be still same.

for example:

In both cases net flow is 1.

∴ If we reduce $f(v,u)$ by $f(u,v)$ and set $f(u,v)=0$, flow value will be still same.

$\therefore$ There always exists a maximum flow f in which either $f(u,v)=0$ or $f(v,u)=0$.