1. (1)

```
IsValidSequence(string S, string sub, int ptr)
    if dict(sub) is true:
print(sub)  sub = []
    sub.append(S[ptr])
    ptr++
    if ptr is len(S):
        if dict(sub) is true:
            return true
        else:
            return false

if "__name" = "__main__":
    IsValidSequence(S, [], 0)
```

(2) see changes in BLUE.

2. Let $M(i,t)$ be true if there is a subset of $\{a_1, a_2, a_3, \cdots, a_i\}$ with summation of t, and be false otherwise.

$$\begin{cases} M(i,t) = M(i-1,t) \cup M(i,t-a_i) & \text{if } t \geq a_i \\ M(i,t) = M(i-1,t) & \text{if } t < a_i \end{cases}$$

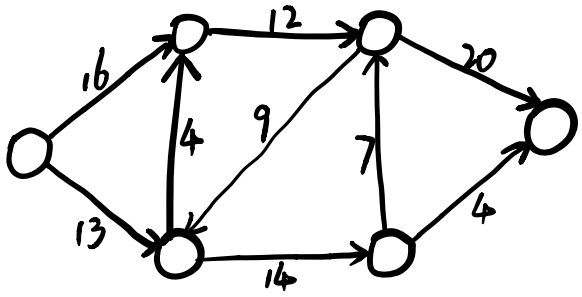
the final answer is given in $M(i,t)$.

3. Assume $E(i,j)$ is the max-scoring alignment of the substring of length of the first string.

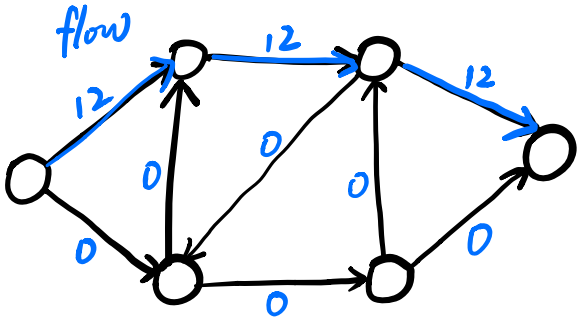
$$E(i,j) \begin{cases} 0, & i=j=0 \\ E(i-1,0) + \delta(x_i, -) & j=0 \\ E(0,j-1) + \delta(-, y_i) & i=0 \\ \max \begin{Bmatrix} E(i-1,j) + \delta(x_i,-) \\ E(i,j-1) + \delta(-,y_i) \\ E(i-1,j-1) + \delta(x_i,y_i) \end{Bmatrix} & \text{otherwise.} \end{cases}$$

The final answer is given in $E(i,j)$.

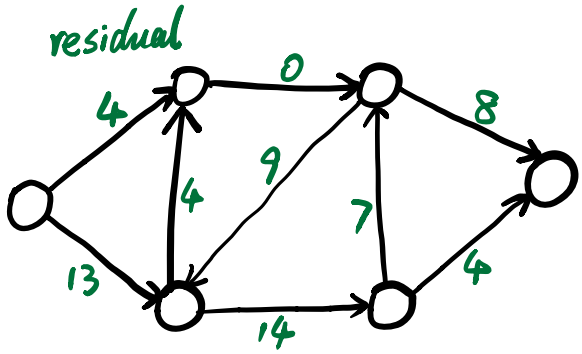
4.



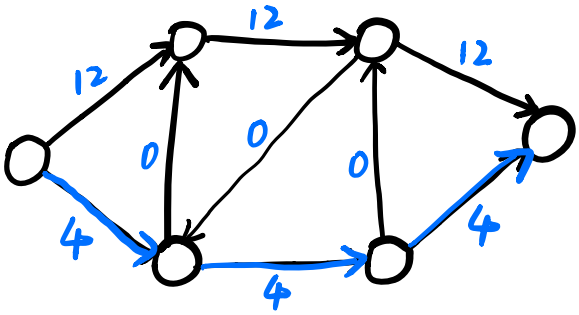
STEP 1.

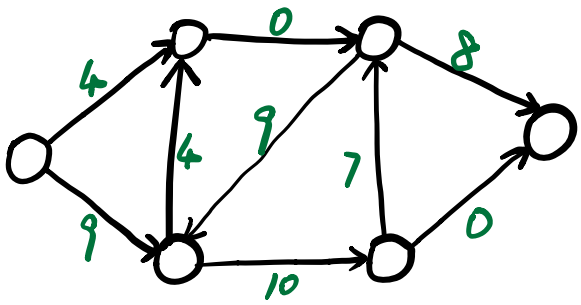


residual

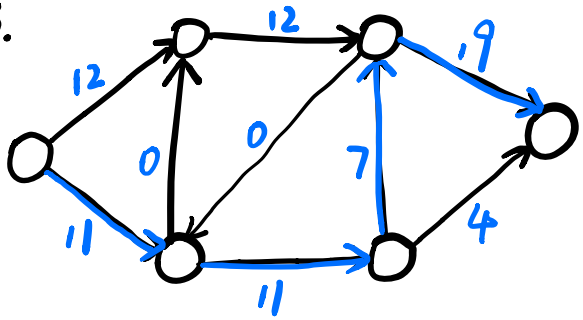


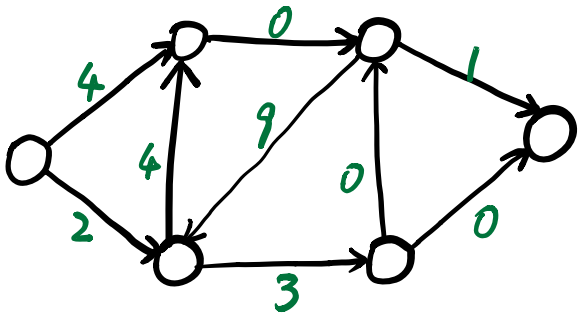
STEP 2.





STEP 3.





No augmenting path anymore.

final graph is step 3 flow graph.

5. Using dynamic programming method.

First. select a node near source and get the max-flow.

Then. choose another node and get the max-flow from source to this node.

⋮ Iterate last step, until got the max-flow from source to sink.

The basic idea is to make local optimal and enlarge this local area, when the area equals to the total graph, then we have the solution we need.