2011/09/01 김석현

1.

a) $29 = 1 + (4\times3) + 2 + (3\times4) + 2$

b) $OPT[j] = \begin{cases} \max\{OPT[j-1]+v_j,\ OPT[j-2]+v_j\times v_{j-1}\} & \text{if } j\geq 2 \\ v_1 & \text{if } j=1 \\ 0 & \text{if } j=0 \end{cases}$

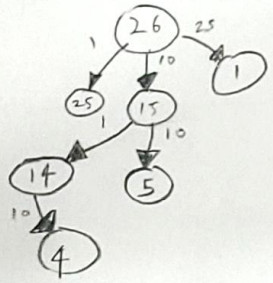
c)
```
Prod-Sum (int[]v, n)
if(n==0) return 0
OPT[0]=0;
OPT[1]=V[1];
for int j=2 to n
    OPT[j]=max(OPT[j-1]+V[j], OPT[j-2]
    + V[j]×V[j-1]); return OPT[n];
```

2. i) n=0 → return 0

ii) $OPT(n) = \begin{bmatrix} OPT(n,1) \\ OPT(n,10) \\ OPT(n,25) \end{bmatrix} + 1$

iii) opt(n) in $n\leq 0$ return n

ex) n=26



3. 만약 14원이 필요하면

1개의 11원, 3개의 1원으로 4개의 동전이 필요하다

2개의 7원이 있으면 더 좋은 방법이 된다

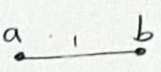
그리드 알고리즘 항상 가장 적은 수의 동전을 산출하는것이 아니다.

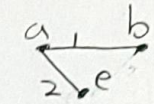
4.

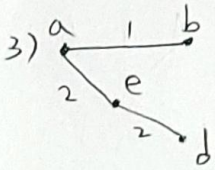
2	2	7	10
3	6	13	15
4	6	16	24
12	14	20	25
13	21	26	31

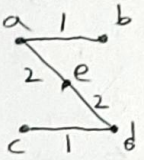
```
DP[0][0]=M[0][0]
for(int i=1; i<M; i++)
DP[0][1]=DP[0][i-1]+M[0][i]
for(int i=1; i<N; i++)
DP[i][0]=DP[i-1][0]+M[i][0]
for(int i=1; i<M; i++)
for(int j=1; j<N; j++)
DP[j][i]=Max(DP[j-1][i]
+M[j][1], DP[j][i-1]+M[j][i]
DP[M-1][N-1]
```

5.

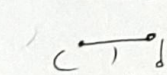
a) 1) 

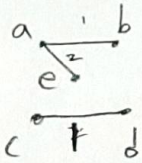
2) 

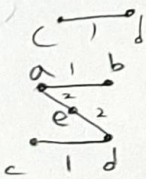
3) 

4) 

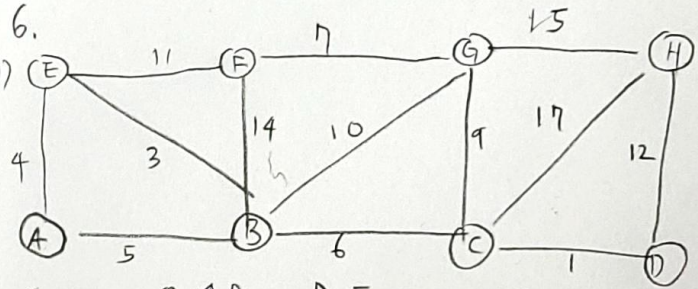
b) 1)

2) 

3) 

4) 

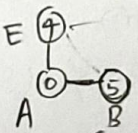
6.

1) 

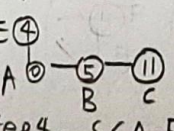
S<> Q<A,B,C,D,E,F,G,H>

step1 - S<A,B> Q<C,D,E,F,G,H>

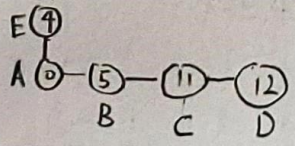
step2 - S<A,E,B> Q<C,D,F,G,H>



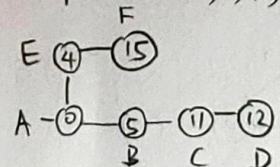
step3 - S<A,E,B,C> Q<D,F,G,H>



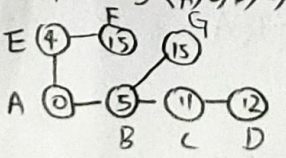
step4 - S<A,E,B,C,D> Q<F,G,H>



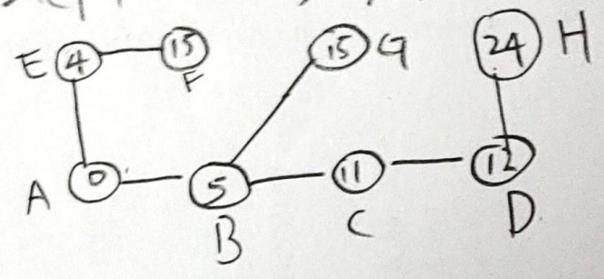
step5 - S<A,E,B,C,D,F> Q<G,H>



step6 - S<A,E,B,C,D,F,G> Q<H>



step7 - S<A,E,B,C,D,F,G,H> Q< >



7. GdC Gb 이다

두 알고리즘 모두 최단 경로를 푸는 알고리즘이지만 bellman-ford 알고리즘에서 변의 가중치는 음수일 수도 있기 때문에 GdC Gb 이다