#1

1. Consider sequence of weights 2,3,2. The algorithm will pick the middle node while the maximum is first and third.
2. Consider the sequence of weights 3 1 2 3. The algorithm will pick the first and thirds while the maximum is first and fourth.
3. Let Si denote and independent set on {v1-vi}, let Xi denote its weight. X0 =0 and X1 = w1. Now for i>1, either vi belongs to Si or not. In the first case, we know that vi-1 cannot belong to Si so Xi = wi+Xi-2. In the second case, Xi =Xi-1. Thus we have Xi=max(Xi-1,wi+Xi-2). Thus we can compute Sn by tracing back through the computations of the max operator. Since we spend constant time per iteration, over N iterations, the total running time is O(n).

#2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Week1 | Week2 | Week3 |
| L | 2 | 2 | 2 |
| H | 1 | 5 | 10 |

The algorithm would take high-stress job in week 2 while the optimal solution take high-stress in week3.

1. Let OPT(i) denote the max value achievable. The optimal solution will select some job in week I, since it’s not worth skipping all jobs. If it selects a high-stress job, it can optimally up to week i-1, followed by this job, while if it selects a high-stress job, it can behave optimally up to week i-2.So OPT(i)=MAX(li+OPT(i-1),hi+OPT(i-2)). The total time is O(n).

#3

1. the graph on nodes v1 – v5 with edges is an example. The algorithm will return 2 while the optimum is 3.
2. By using dynamic programming, denote OPT[i] for the length of the longest path from v1 to vi. Use the value “-inf” for OPT[i] value

Long-path(n)

Arra M[1 to n]

M[1]=0

For i=2 to n

M=-inf

For all edges(j,i) then

If M[j] !=-inf

If M<M[j]+1

M=M[j]+1

Endif

Endif

Endif

M[i] = M

Endfor

Return M[n]

The running time is O(n^2)

#4

1. if M=10, {N1,N2,N3}={1,4,1}, {S1,S2,S3} = {20,1,20} the optimal is [NY,NY,NY] while the greedy algorithm is [Ny,SF,NY]
2. if M=10, {N1,N2,N3,N4}={1,100,1,100},{S1,S2,S3,S4} = {100,1,100,1}, the result is [Ny,SF,NY,SF] which is higher than other plan.
3. The optimal plan either ends in NY, or SF. If it ends in NY, it will pay Nn+the cost of the optimal plan on n-1 months ending in NY or the cost of optimal on n-1 months ending in SF plus a moving cost M. So OPTN(n) = Nn + min(OPTN(n-1)),M+OPTS(n-1)); OPTS(n) = Sn + min(OPTS(n-1)),M+OPTN(n-1))🡪OPTN(0) = OPTS(0) =0

for i=1 to n

OPTN(i) = Ni + min(OPTN(i-1)),M+OPTS(i-1))

OPTS(i) = Si + min(OPTS(i-1)),M+OPTN(i-1))

End

Return Min(OPTN(n),OPTS(n)); the running time is O(n)

#5

let OPT(i) be the score of the best segmentation of the prefix consisting of the first I characters of y. We claim that the recurrence OPT(i) = min(i<=i){OPT(j-1)+Quality(j…n)}; the running time is quadratic time.

#6

define OPT(i) to be the value of the optimal solution on the set of words Wi = {w .. wi}. For any i<=j, let Si,j denote the slack of the line containing the words wi,,,wj; OPT(n) = min(1<=j<=n){S(i,n)^2 +OPT[j-1]

Compute all values Si,j as described above

Set OPT(0) = 0

For k = 1 to n calculate OPT

Return OPT(n)

The algorithm will take O(n^2)

#7

let Xj(j=1 to n) denote the max possible return the investors can make if they sell the stock on day j. then Xj = max(0, Xj-1 +(p(j)-p(j-1))). The answer is the maximum over j = 1 to n of Xj.

#8

1. Change x4 to 2. The algorithm would activate the EMP at times 2 and 4, while activate at times 3 and 4 as before still gets 5.
2. Let OPT(i) be the max number of robots that can be destroyed. OPT(j) = max(0<=i<j){OPT(i)+min(xi,f(j-i))}

Set OPT(0) =0

For I = 1 to n

Compute OPT(j)

End for

Return OPT(n)

The running time is O(n^2)

#9

1. suppose s1 =10 and si =1 for all i>1; and xi=11 for all i. The optimal solution should reboot in every other day, thereby processing 10 terabytes every two days.
2. Let OPT(i) denote the max number of bytes that can be processed from day 1 to day i.the total number of bytes processed will be b(ij) = sum(k=1|i-j)(min{Xj+k,sk}) the total work processed till day I would be Opt(j-1) + bji. So OPT(i) = max(j=0|i-1)(OPT(j)) +bji.

running time is O(n^3)

#10

|  |  |  |
| --- | --- | --- |
|  | Minute1 | Minute2 |
| A | 2 | 10 |
| B | 1 | 20 |

The greedy algorithm would chose A for both steps while the optimal solution would be choose B for both steps.

Let OPT(I,A)denote the max value of a plan in minutes 1 through I that ends on machine A, and OPT(I,B) analogously for B. So we have OPT(I,A) = ai +max(OPT(i-1,A),OPT(i-1,B)). The running time is O(n).