Week 7 Solution

1.You are given two implementations for finding the nth Fibonacci number(F)
Fibonacci numbers are defined by
F(n) = F(n-1) + F(n-2)
with F(0) = 0 and F(1) = 1

The two implementations are
1. Approach 1

```
int fib(int n)
{
  if (n <= 1)
     return n;
  return fib(n-1) + fib(n-2);
}
```

2. Approach 2

```
int fib(int n)
{
 /* array to store fibonacci numbers. */
 int f[n+1];
 int i;
 f[0] = 0;
 f[1] = 1;
 for (i = 2; i <= n; i++) {

   f[i] = f[i-1] + f[i-2];
 }
return f[n];
}
```

Which of the two algorithms has better time complexity ?

1. Approach 1
2. Approach 2
3. Both have same time complexity

Answer: 2
Explanation: Approach 2 is linear time while approach 1 is exponential time

2.Consider the problem of matrix chain multiplication.

Let $p_0,p_1,p_2,....p_n$ be the dimension of the matrices
$A_1\ A_2\ ....\ A_n$ such that dimension of A_i is p_{i-1} x p_i

Given the structure of optimal solution as
$Ai...j = (A_i\ ....\ A_k)(A_{k+1}\ ....\ A_j)$

For $1 \le i \le j \le n$, Let m[i,j]
denote the minimum number of multiplications needed to compute Ai...j . The optimum cost can be described by which of the following recursive definition

A. $m[i,j] = \min_{i \le k < j}(m[i,k] + m[k+1,j] + p_{i-1}\ p_k\ p_j)$
B. $m[i,j] = \min_{i \le k \le j}(m[i,k] + m[k+1,j] + p_{i-1}\ p_k\ p_j)$
C. $m[i,j] = \min_{i \le k < j}(m[i,k] + m[k,j] + p_{i-1}\ p_k\ p_j)$
D. $m[i,j] = \min_{i \le k < j}(m[i,k-1] + m[k+1,j] + p_{i-1}\ p_k\ p_j)$

Answer: A

3. Consider the following 4 matrices

A : 5 x 4
B : 4 x 6
C : 6 x 2
D : 2 x 7

Using the recursive definition for matrix chain multiplication, compute the values for x and y in the table below

	1	2	3	4
1	0	x	y	
2	x	0	48	104
3	y	48	0	84
4		104	84	0

A. x = 88 and y = 120
B. x = 116 and y = 88
C. x = 120 and y = 116
D. x = 120 and y = 88

Answer: D

4.What is the total number of scalar multiplications required to multiply the 4 matrices defined in question above?

A. 36
B. 168
C. 158
D. 104

Answer : C
Explanation: Answer can be verified by filling the table using the recursive definition.

5. int fun(int n){
T[0]=T[1]=2;
T[2]=2*T[0]*T[1];

```
for (int i=3;i<n;i++)
   T[i]=T[i-1]+2*T[i-1]*T[i-2];
return T[n]
}
```

Which of the following corresponds to the space and time complexity for the above code ?

A. O(n) & O(n)
B. O(n) & $O(n^2)$
C. $O(n^2)$ & O(n)
D. None of the above

Answer: A

```
6. int fun(int n){
T[0]=T[1]=2;
T[2]=2*T[0]*T[1];
for (int i=3;i<n;i++)
   T[i]=T[i-1]+2*T[i-1]*T[i-2];
return T[n]
}
```

If T[0]=T[1]=2, then for n>1 the recurrence relation for the above code can be given by:
T[0]=T[1]=2.

A. i=1n-12*T[i]*T[i-1]
B. i=1n-12*T[i-1]*T[i-2]
C. 2*T[i]*T[i-1]

i=1n2*T[i]*T[i-1]

7.Given n types of coin denominations of values $V_1<V_2<\ldots<V_n$(integers).
Let V_1=1, so that for any amount of money M change can always be made.
If T(j) indicates the minimum number of coins requires to make a change for the amount of money equal to j and coin V_k was the last denomination added to the solution.

Then which recurrence best describes T(j) ?

A. $Min_k\{T(j-V_k)\}+1$
B. $Min_k\{T(j-V_k)\}$
C. $T(j-V_k)+1$
D. None of the above

Answer: A.
Explanation:

if the coin denomination k was the last denomination added to the solution then the optimal way to finish the solution with that one is optimally make change for the amount of money $j-v_k$ and then add one extra coin of value V_k ?

8. Consider the following set of denominations {1,3,4,5} in a currency.What is the number of coins returned by the greedy strategy to make change for 7 rupees ?
Also state what is the number of coins in an optimal solution?

A. 3, 2
B. 2, 3
C. 3, 3
D. None of the above

Answer: A.
Explanation:
Using the greedy strategy we'll get one coin of denomination 5 and 2 coins of denomination 1.
The best solution would be to pick one coin of denomination 3 and one coin of denomination 4

9. We use dynamic programming approach when

A. The solution has optimal substructure
B. The problem can be has divided into subproblems which are overlapping.
C. The problem can be has divided into subproblems and a global solution can be achieved by making locally optimal solutions
D. A & C
E. A & B

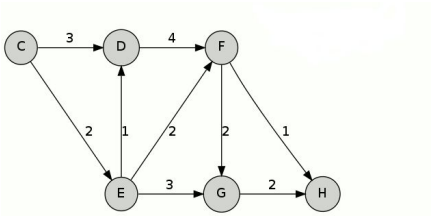
Answer: E
Explanation: DP needs optimal substructure and overlap between subproblems.

10. We use Greedy approach when

A. The solution has optimal substructure
B. The problem can be has divided into subproblems which are overlapping.
C. The problem can be has divided into subproblems and a global solution can be achieved by making locally optimal solutions
D. A & C
E. A & B

Answer: D
Explanation: Greedy needs optimal substructure and global solution should be obtainable by making locally optimal solutions.

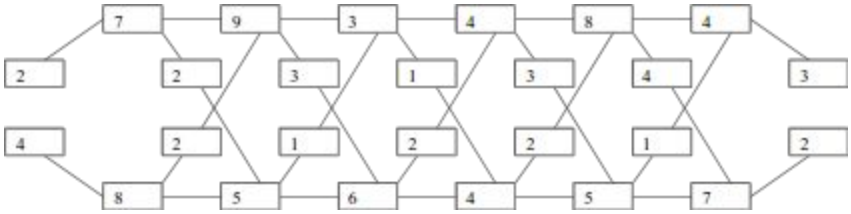
11. Find length of Shortest path from C to H?



A. 8
B. 5
C. 7
D. 4

Answer: B
Explanation: C->E->F->H

12.



Given above assembly line with 6 stations, find fastest time to get through entire factory

A. 38
B. 36
C. 39
D. None of the above

Answer: A
Explanation:

i \ j	1	2	3	4	5	6	exit
$f_1[j]$	1: 2+7 = **9**	**1:** 9 +9 = **18** 2: 12+2+9 = 21	1: 18 +3 = 21 **2:** 16+1+3 = **20**	**1:** 20 +4 = **24** 2: 22+2+4 = 28	**1**: 24 +8 = **32** 2: 25+2+8 = 35	1: 32 +4 = 36 **2**: 30+1+4 = **35**	35+3 = **38**
$f_2[j]$	2: 4+8 = **12**	2: 12+ 5 = 17 **1:** 9+2+5 = **16**	**2:** 16 +6 = **22** 1: 18+3+6 = 27	2: 22 +4 = 26 **1:** 20+1+4 = **25**	**2**: 25 +5 = **30** 1: 24+3+5 = 32	**2**: 30 +7 = **37** 1: 32+4+7 = 43	37+2 == 39

13. A naive way to calculate the nth Fibonacci number is to use the definition of Fibonacci number
F(n) = F(n-1) + F(n-2), F(0) = F(1) = 1
We know that this algorithm is exponential, because we do a lot of repetitive computation.
A DP solution to the above problem would give us a linear time algorithm. The number of calculations done by naive method in computing F(4) are ?

Answer: 9
Explanation:

F(4) = F(3) + F(2)
F(3) = F(2) + F(1)
F(2) = F(1) + F(0)

F(4) = 8 calls in calls + 1 call to F(4)

14. For the coin change problem, does greedy algorithm design strategy always result in the right answer?

A. Yes
B. No

Answer: B
Explanation: Optimality of greedy strategy is not guaranteed for arbitrary basic coins.
For example:
If basic coins were (1,3,4)
To get coins worth 6
Greedy would give (4,1,1)
while optimal would be (3,3)