CX2101 Algorithm Design and Analysis

Tutorial 4: Dynamic Programming

School of Computer Science and Engineering *Nanyang Technological University*

Week 10 (Q1-Q3):

1. Find the length of the longest common subsequence and a longest common subsequence of CAGAG and ACTGG by the dynamic programming algorithm in the lecture notes.

2. The H-number H(n) is defined as follows:

H(0) =1, and for n>0:
H(n) = H(n-1) + H(n-3) + H(n-5)+ ….+H(0) when n is odd
H(n) = H(n-2) + H(n-4) + H(n-6)+ ….+H(0) when n is even.

a) Give a recursive algorithm to compute H(n) for an arbitrary n as suggested by the recurrence equation given for H(n). Draw the tree that represents the recursive calls made when H(8) is computed.
b) Draw the subproblem graph for H(8) and H(9).
c) Write an iterative algorithm using the dynamic programming approach (bottom-up). What are the time and space required?

3. The binomial coefficients can be defined by the recurrence equation:

$C(n, k) = C(n - 1, k - 1) + C(n - 1, k)$	for $n > 0$ and $k > 0$
$C(n, 0) = 1$	for $n >= 0$
$C(0, k) = 0$	for $k > 0$

C(n, k) is also called "n choose k". This is the number of ways to choose k distinct objects from a set of n objects.

(a) Give a recursive algorithm as suggested by the recurrence equation given for C(n, k).
(b) Draw the subproblem graph for C(5, 3).
(c) Write a recursive algorithm using the dynamic programming approach (top-down) stating the data structure used for the dictionary. What is the space and time complexity respectively?
(d) Write an iterative algorithm using the dynamic programming approach (bottom-up). What is the space and time complexity respectively?

Week 11 (Q4-Q6):

4. Suppose the dimensions of the matrices A, B, C, and D are 20x2, 2x15, 15x40, and 40x4, respectively, and we want to know how best to compute AxBxCxD.

Week 10

1. Find the length of the longest common subsequence and a longest common subsequence of CAGAG and ACTGG by the dynamic programming algorithm in the lecture notes.

c:

		A	C	T	G	G
	0	0	0	0	0	0
C	0	0	1	1	1	1
A	0	1	1	1	1	1
G	0	1	1	1	2	2
A	0	1	1	1	2	2
G	0	1	1	1	2	3

h:

		A	C	T	G	G
	—	—	—	—	—	—
C	|	|	\	—	—	—
A	|	\	|	|	|	|
G	|	|	|	|	\	\
A	|	\	|	|	|	|
G	|	|	|	|	\	\

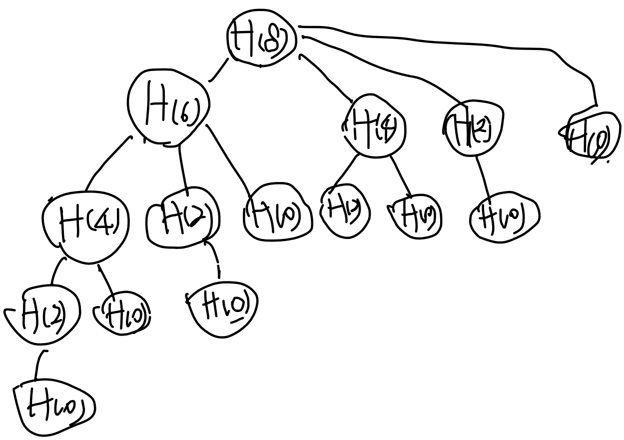
∴ ANS : CGG

2. The H-number H(n) is defined as follows:

H(0) =1, and for n>0:
H(n) = H(n-1) + H(n-3) + H(n-5)+ ….+H(0) when n is odd
H(n) = H(n-2) + H(n-4) + H(n-6)+ ….+H(0) when n is even.

a) Give a recursive algorithm to compute H(n) for an arbitrary n as suggested by the recurrence equation given for H(n). Draw the tree that represents the recursive calls made when H(8) is computed.

a)
```
int H(int n){
  if (n==0) //return base value!
     return 1
  int result = 0;
  while (n>0){
     if (n%2==1){
        result += H(n-1);
        n--;
     else{
        result += H(n-2);
        n-=2;
  }
  return result;
```

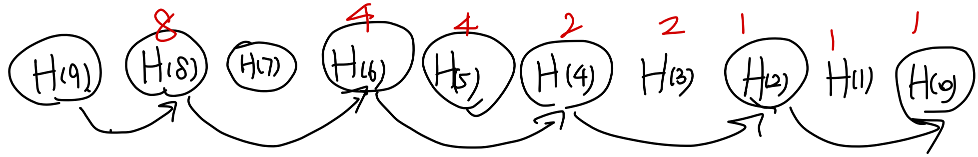


2. The H-number H(n) is defined as follows:

H(0) =1, and for n>0:
H(n) = H(n-1) + H(n-3) + H(n-5)+ ….+H(0) when n is odd
H(n) = H(n-2) + H(n-4) + H(n-6)+ ….+H(0) when n is even.

a) Give a recursive algorithm to compute H(n) for an arbitrary n as suggested by the recurrence equation given for H(n). Draw the tree that represents the recursive calls made when H(8) is computed.
b) Draw the subproblem graph for H(8) and H(9).
c) Write an iterative algorithm using the dynamic programming approach (bottom-up). What are the time and space required?

b)



c)

```
int H (int n) {
    int* sum = new int [n+1];
    if (n ≤ 2)
        return 1;
    H[0] = H[1] = H[2] = 1;
    int lst = n%2 == 0 ? n-1 : n;
    for (int i = 3; i ≤ lst ; i += 2)
        sum[i] = sum[i-2] x 2 ;
    return sum[lst];
}
```

Time & Space : O(n)

3. The binomial coefficients can be defined by the recurrence equation:

$C(n, k) = C(n-1, k-1) + C(n-1, k)$	for $n > 0$ and $k > 0$
$C(n, 0) = 1$	for $n >= 0$
$C(0, k) = 0$	for $k > 0$

$C(n, k)$ is also called "n choose k". This is the number of ways to choose k distinct objects from a set of n objects.

(a) Give a recursive algorithm as suggested by the recurrence equation given for $C(n, k)$.
(b) Draw the subproblem graph for $C(5, 3)$.
(c) Write a recursive algorithm using the dynamic programming approach (top-down) stating the data structure used for the dictionary. What is the space and time complexity respectively?
(d) Write an iterative algorithm using the dynamic programming approach (bottom-up). What is the space and time complexity respectively?

(a)

Show the arrays **cost**, **last**, and **multOrder** computed by Algorithms matrixOrder() in the lecture notes.

5. Construct an example with only three or four matrices where the worst multiplication order does at least 100 times as many element-wise multiplications as the best order.

6. We have a knapsack of size 10 and 4 objects. The sizes and the profits of the objects are given by the table below. Find a subset of the objects that fits in the knapsack that maximizes the total profit by the dynamic programming algorithm in the lecture notes.

p	10	40	30	50
s	5	4	6	3