Problem Statement: Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyse their time and space complexity.

jupyter Untitled Last Checkpoint: 12 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[6]:
```python
nterms = int(input("Number of terms"))
F1=0
F2=1
count=1
if nterms<=0:
    print("Please enter a positive integer")
elif nterms==1:
    print("Fibonacci sequence is:")
    print("F",nterms,":",F1)
else:
    print("Fibonacci sequence is:")
    while count<=nterms:
        print(F1)
        Fth=F1+F2
        F1=F2
        F2=Fth
        count+=1
```

```
Number of terms 10
Fibonacci sequence is:
0
1
1
2
3
5
8
13
21
34
```

[ ]:

jupyter Untitled Last Checkpoint: 23 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
34
```

[13]:
```python
def fib_rec(n):
    if n<=1:
        return n
    else:
        return(fib_rec(n-1)+fib_rec(n-2))
nterm = int(input("Enter the number of terms:"))
if nterm<=0:
    print("Enter positive integer")
else:
    print("Fibonacci Sequence:")
    for i in range(nterm):
        print("F",i,":",fib_rec(i))
```

```
Enter the number of terms: 4
Fibonacci Sequence:
F 0 : 0
F 1 : 1
F 2 : 1
F 3 : 2
```

[ ]:

Problem Statement: Write a program to implement Huffman Encoding using a greedy strategy.

McAfee Security Dell YouTube Road To Code LinkedIn github Netlify CloudyML Other favorites

Code Open in... Python 3 (ipykernel)

[1]:

```python
import heapq

# Creating Huffman tree node
class node:
    def __init__(self,freq,symbol,left=None,right=None):
        self.freq=freq # frequency of symbol
        self.symbol=symbol # symbol name (character)
        self.left=left # node left of current node
        self.right=right # node right of current node
        self.huff= '' # # tree direction (0/1)

    def __lt__(self,nxt): # Check if curr frequency less than next nodes freq
        return self.freq<nxt.freq

def printnodes(node,val=''):
    newval=val+str(node.huff)
    # if node is not an edge node then traverse inside it
    if node.left:
        printnodes(node.left,newval)
    if node.right:
        printnodes(node.right,newval)

    # if node is edge node then display its huffman code
    if not node.left and not node.right:
        print("{} -> {}".format(node.symbol,newval))

if __name__=="__main__":
    chars = ['a', 'b', 'c', 'd', 'e', 'f']
    freq = [ 5, 9, 12, 13, 16, 45]
    nodes=[]
```

McAfee Security Dell YouTube Road To Code LinkedIn github Netlify CloudyML Other favorites

Code Open in... Python 3 (ipykernel)

```python
    # if node is edge node then display its huffman code
    if not node.left and not node.right:
        print("{} -> {}".format(node.symbol,newval))

if __name__=="__main__":
    chars = ['a', 'b', 'c', 'd', 'e', 'f']
    freq = [ 5, 9, 12, 13, 16, 45]
    nodes=[]

    for i in range(len(chars)): # converting characters and frequencies into huffman tree nodes
        heapq.heappush(nodes, node(freq[i],chars[i]))

    while len(nodes)>1:
        left=heapq.heappop(nodes)
        right=heapq.heappop(nodes)

        left.huff = 0
        right.huff = 1
        # Combining the 2 smallest nodes to create new node as their parent
        newnode = node(left.freq + right.freq , left.symbol + right.symbol , left , right)
        # node(freq,symbol,left,right)
        heapq.heappush(nodes, newnode)

    printnodes(nodes[0]) # Passing root of Huffman Tree
```

```
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
```

Problem Statement: Write a program to solve a fractional Knapsack problem using a greedy method.

```
# Knapsack problem using greedy method
class Item:
    def __init__(self,value,weight):
        self.value=value
        self.weight=weight
def fracKnapsack(W,arr):
    arr.sort(key=lambda x:(x.value/x.weight),reverse=True)
    finalvalue=0.0
    for i in arr:
        if i.weight<=W:
            W-=i.weight
            finalvalue+=i.value
        else:
            finalvalue+=i.value*W/i.weight
            break
    return finalvalue



if __name__=="__main__":
    W=50
    arr=[Item(60,10),Item(100,20),Item(120,30)]

    max_val=fracKnapsack(W, arr)
    print(max_val)
```

```
240.0
```

Problem Statement: Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.

jupyter Untitled1 Last Checkpoint: 11 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code Open in... Python 3 (ipykernel)

```
e -> 111
```

[2]:

```
def solve_knapsack():
    val=[50,100,150,200] #value array
    wt=[8,16,32,40] # Weight array
    W=64
    n=len(val) - 1
    def knapsack(W,n): # (Remaining Weight, Number of items checked)
        #base case
        if n<0 or W<=0:
            return 0

        #Higher weight than available
        if wt[n]>W:
            return knapsack(W, n-1)

        else:
            return max(val[n] + knapsack(W-wt[n],n-1),knapsack(W,n-1))
            # max(including , not including)
    print(knapsack(W,n))

if __name__=="__main__":
    solve_knapsack()
```

```
350
```

Problem Statement: Design n-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final n-queen's matrix.

jupyter Untitled1 Last Checkpoint: 13 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code | Open in... | Python 3 (ipykernel)

[3]:
```python
def n_queens(n):
    col = set()
    posDiag=set() # (r+c)
    negDiag=set() # (r-c)

    res=[]

    board = [["0"]*n for i in range(n) ]
    def backtrack(r):
        if r==n:
            copy = [" ".join(row) for row in board]
            res.append(copy)
            return

        for c in range(n):
            if c in col or (r+c) in posDiag or (r-c) in negDiag:
                continue

            col.add(c)
            posDiag.add(r+c)
            negDiag.add(r-c)
            board[r][c]="1"

            backtrack(r+1)

            col.remove(c)
            posDiag.remove(r+c)
            negDiag.remove(r-c)
            board[r][c]="0"
```

McAfee Security | Dell | YouTube | Road To Code | LinkedIn | github | Netlify | CloudyML | Other favorites

jupyter Untitled1 Last Checkpoint: 13 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code | Open in... | Python 3 (ipykernel)

```python
            board[r][c]="0"
    backtrack(0)
    for sol in res:
        for row in sol:
            print(row)
        print()

if __name__=="__main__":
    n_queens(8)
```

```
1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0

1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
```

[ ]:

// Write a program to implement matrix multiplication. Also implement multithreaded matrix multiplication with either one thread per row or one thread per cell. Analyze and compare their performance.

```
#include <bits/stdc++.h>

using namespace std;

#define MAX 4

#define MAX_THREAD 4


int matA[MAX][MAX];

int matB[MAX][MAX];

int matC[MAX][MAX];

int step_i = 0;


void* multi(void* arg)

{

   int i = step_i++;


   for (int j = 0; j < MAX; j++)

   for (int k = 0; k < MAX; k++)

      matC[i][j] += matA[i][k] * matB[k][j];

}


int main()

{

   for (int i = 0; i < MAX; i++) {

      for (int j = 0; j < MAX; j++) {

         matA[i][j] = rand() % 10;

         matB[i][j] = rand() % 10;

      }

   }

   cout << endl

      << "Matrix A" << endl;

   for (int i = 0; i < MAX; i++) {

      for (int j = 0; j < MAX; j++)
```

```
            cout << matA[i][j] << " ";
        cout << endl;
    }
    cout << endl
        << "Matrix B" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matB[i][j] << " ";
        cout << endl;
    }

    pthread_t threads[MAX_THREAD];



    for (int i = 0; i < MAX_THREAD; i++) {
        int* p;
        pthread_create(&threads[i], NULL, multi, (void*)(p));
    }
    for (int i = 0; i < MAX_THREAD; i++)
        pthread_join(threads[i], NULL);

    cout << endl
        << "Multiplication of A and B" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matC[i][j] << " ";
        cout << endl;
    }
    return 0;
}
// Time Complexity: O(1)
// Auxiliary Space: O(1)
```

```
                Matrix Multiplication Using Threads
-------------------------------------------------------------------
        Note: The thread number would be given the value of p
         so enter the value according to available resources.
Rows and columns of matrix A are p & q
Rows and columns of matrix B are q & r

Enter number of rows (p) & number of columns (q) of matrix A: 3
3
Enter number of columns (r) of matrix B: 4
Enter Y/y to enter values to both matrices else random values will be generated: n

Matrix A:
3 6 7
5 3 5
6 2 9

Matrix B:
1 2 7 0
9 3 6 0
6 2 6 1

Product matrix C is:
99 38 99 7
62 29 83 5
78 36 108 9
```