ADS HW2 Report

404261476 資工四甲 楊培澤

Problem : Optimal Binary Search Tree

Solution :

Using dynamic programming to solve this problem. First, count the cost for every possible path. Then use dynamic programming to get the optimal path and the the total cost.

Discussion :

Although I perform dynamic programming to solve this problem, I think there might be a better solution to this problem. Need some advice to implement a better algorithm.

Code :

```
if __name__ == '__main__':
    # dir
    input_dir = './data/Optimal Binary Search Tree.in'
    output_dir = './data/Optimal Binary Search Tree.out'


    # load all data
    data = [line.strip('\r').strip('\n').split(' ') for line in open(input_dir, 'r').readlines()]

    # doc
    ans = []
```

```
        # get OBST in each row
        for idx in range(len(data)):
            item = data[idx]
            result = [[1e-2 for i in range(252)] for j in range(252)]
            cost = [[1e-2 for i in range(252)] for j in range(252)]
            n = int(item[0])

            # init result diagonal
            for i in range(1, n+1):
                result[i][i] = 0

            # count size
            sum_arr = [0]
            for i in range(1, n+1):
                sum_arr.append(sum_arr[len(sum_arr)-1] + int(item[i]))
            # print('sum_arr', sum_arr)

            # count cost
            for i in range(1, n+1):
                for j in range(i, n+1):
                    cost[i][j] = sum_arr[j] - sum_arr[i-1]

            # list init
            for i in range(1, n+1):
                result[i][i-1] = 0
                cost[i][i-1] = 0
            result[n+1][n] = 0
            cost[n+1][n] = 0



    # size dynamic programming
    for tmp in range(1, n):
        for i in range(1, n-tmp+1):
            j = tmp + i
            result[i][j] = 1e9
            for k in range(i, j+1):
                if result[i][k - 1] + result[k + 1][j] + cost[i][k-1] + cost[k+1][j] < result[i][j]:
                    result[i][j] = result[i][k - 1] + result[k + 1][j] + cost[i][k-1] + cost[k+1][j]
    ans.append(result[1][n])
    print('\r[%d/%d] Processing...' % (idx+1, len(data)), end='')
print('\rFinished...Wait for evaluate')
```

```
# check accuracy
result = ans
error = []
ground_truth = open(output_dir, 'r').readlines()
if len(result) != len(ground_truth):
    print('Result count inconsist')
else:
    count = 0
    for i in range(len(result)):
        word = int(ground_truth[i].strip())
        if result[i] == word:
            count += 1
        else:
            error.append({
                'line': i+1,
                'result': result[i],
                'ground_truth': word
            })
    if count == len(result):
        print('Match output file')
    else:
        print('Some error in result')
        print(error)

# make output file
filename = ''.join(['./', 'output.txt'])
with open(filename, 'w') as f:
    for i in range(len(result)-1):
        f.write("%s\n" % result[i])
    f.write("%s" % result[len(result)-1])
print('Output predict file...')
```