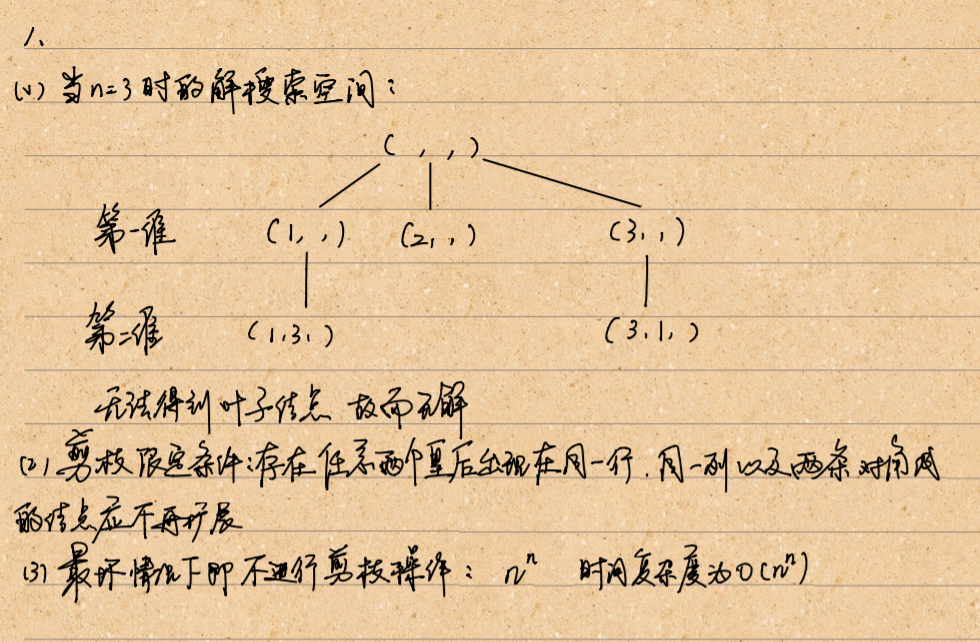
作业2-2018302141224-吴健豪：

李老师您好！这是我的第二次作业

您如果觉得看起来不方便可以查看我的GitHub：

<https://github.com/rushingfox/AlgorithmHomework/tree/master/homework2>

**1.**



**2.**

#include <iostream>

#include <stack>

#include <algorithm>

#include <iterator>

#include <vector>

using namespace std;

vector<int> w = { 1,2,3,4 };

const int n = 4;

const int W = 10;

int Result[n];

int total=0;

bool dfs(int, int, int, int[]);

int main()

{

int temp[n];

for (int i = 0; i < n; i++)

{

total += w[i];

Result[i] = 0;

temp[i] = 0;

}

if (dfs(0,total,0,temp))

{

cout << "the result whose total is equal to " << W << " was as showed below:" << endl;

for (int i = 0; i < n; i++)

{

if (Result[i]==1)

{

cout << "No." <<i<<" item was selected!" << endl;

}

}

}

system("pause");

}

bool dfs(int presentweight,int restweight,int i,int temp[])

{

if (presentweight==W)

{

for (int i = 0; i < n; i++)

{

Result[i] = temp[i];

}

return true;

}

else

{

if (i>=n)

{

//it still does not meet the requirement even it has turned to the end

}

else

{

temp[i] = 1;

if (presentweight+w[i]<=W)//we only preserve the nodes whose weight is still not beyond the aimed number

{

if (dfs(presentweight + w[i], restweight - w[i], i + 1, temp))

{

return true;

}

}

temp[i] = 0;

if (presentweight+restweight-w[i]>=W)//we only preserve the nodes whose weight still has the potential for meeting the number requirements

{

if (dfs(presentweight, restweight - w[i], i + 1, temp))

{

return true;

}

}

}

}

return false;//the final exit for this function

}

**3.**

//题目要求：给定若干个正整数a0、a0 、…、an-1 ，从中选出若干数，使它们的和恰好为k，要求找选择元素个数最少的解。

#include <iostream>

#include <stack>

#include <algorithm>

#include <iterator>

#include <vector>

using namespace std;

int a[] = { 1,4,10,2,6 };

const int totalNum = 5;

const int requiredSum = 7;

int result[totalNum];

int sum = 0;

int minnum = 10000;//represent gigantic

void dfs(int, int, int, int[], int);

int main()

{

int temp[totalNum];

for (int i = 0; i < totalNum; i++)

{

sum += a[i];

result[i] = 0;

temp[i] = 0;

}

dfs(0, 0, sum, temp, 0);

cout << "the result is below:" << endl;

bool find = false;

for (int i = 0; i < totalNum; i++)

{

if (result[i]==1)

{

find = true;

cout << "No. " << i << " was selected!" << endl;

}

}

if (!find)

{

cout << "there is no selection meeting the requirements, not to mention the least num of elements!" << endl;

}

system("pause");

}

void dfs(int i, int presentsum, int restsum, int temp[],int num)//num denotes the number of number in temp

{

if (presentsum==requiredSum && num<minnum)

{

for (int i = 0; i < totalNum; i++)

{

result[i] = temp[i];

}

minnum = num;

}

else

{

if (i>=totalNum)

{

//it still does not meet the requirement even it has turned to the end

}

else

{

temp[i] = 1;

num += 1;

if (presentsum+a[i]<=requiredSum)

{

dfs(i + 1, presentsum + a[i], restsum - a[i], temp, num + 1);

}

temp[i] = 0;

num -= 1;

if (presentsum+restsum-a[i]>=requiredSum)

{

dfs(i + 1, presentsum, restsum - a[i], temp, num);

}

}

}

}

**4.**

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int a[] = { 1,1,2 };

const int n = 3;

void dfs(int, int[]);

bool repeated(int[], int, int);

int main()

{

cout << "all the arrangement was below:";

dfs(0, a);

system("pause");

}

void dfs(int num,int a[])

{

if (num>=n)

{

cout << endl;

//a new arrangement was found and printed

for (int i = 0; i < n; i++)

{

cout << " " << a[i];

}

}

else

{

dfs(num + 1, a); //default route

for (int i = num+1; i < n; i++) //possible routes

{

if (!repeated(a,num,i))

{

int transit = a[num];

a[num] = a[i];

a[i] = transit;

dfs(num + 1, a);

a[i] = a[num];//backtracking

a[num] = transit;

}

}

}

}

bool repeated(int a[],int num,int i)

{

for (int j = num; j < i; j++)

{

if (a[i]==a[j])

{

return true;

}

}

return false;

}

**5.**

#include <iostream>

#include <vector>

#include <iterator>

using namespace std;

const int n = 3;

const int m = 2;

void dfs(int, vector<int>, int[m]);

int main()

{

int temp[m];

for (int i = 0; i < m; i++)

{

temp[i] = 0;

}

vector<int> a;

for (int i = 0; i < n; i++)

{

a.push\_back(i + 1);

}

cout << "the result is below:" << endl;

dfs(0, a, temp);

system("pause");

}

void dfs(int num,vector<int> a,int b[m])

{

if (num>=m)

{

cout << endl;

for (int i = 0; i < m; i++)

{

cout << " " << b[i];

}

}

else

{

for (int i = 0; i < a.capacity(); i++)

{

b[num] = a[i];

a.erase(a.begin() + i);

dfs(num + 1, a, b);

a.insert(a.begin() + i, b[num]);//backtracking

b[num] = 0;

}

}

}