20174179 杨小川49 计算机科学与技术（卓越）

15.1-3我们对钢条切割问题进行了一点修改。除了切割下的钢条段具有不同的价格pi之外，每次切割还要付出固定的成本c，这样，切割方案的收益就等于钢条段的价格之和减去切割的成本。设计一个动态规划算法解决修改之后的钢条切割问题。

自下而上的方法：

伪代码：

BOTTOM\_UP\_CUT ROD(p,n)

Let r[0..n]be a new array

For j = 1 to n

q = max(p[0]+r[j], p[j])

For i = 1 to j

q = max(q,p[i]+r[j-i]-ExtraFee)

r[j] = q

Return r[n]

C++实现：

#include <iostream>

#define UNV -9999*//* *定义为负无穷*

#define EXFEE 2*//* *每次切割产生的额外费用*

using namespace std;

int BOTTOM\_UP\_CUT\_ROD(int p[],int n)

{

int \*r=new int[n];

r[0]=0;

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

{

int q = Max(p[0]+r[j], p[j]+r[0]);*//若不切则不产生额外费用*

for (int i=1;i<j;i++)

{

q = Max(q, p[i]+r[j-i]-EXFEE);*//若切割则每次产生额外费用*

}

r[j+1]=q;

}

int result = r[n];

delete []r;

return result;

}

int main(int argc, char \*argv[])

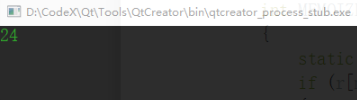
{

int arr[10] = {1, 5, 8, 9, 10, 17, 17, 20, 24, 30};

printf("%d", BOTTOM\_UP\_CUT\_ROD(arr, 9));

return 0;

}



利用辅助数组的带备忘的自顶而下的方法：

伪代码：

MEMOIZED\_CUT\_ROD(p[], n)

Let r[0..n]be a new array

For i = 0 to n

r[i] = -∞

Return MEMOIZZED\_CUT\_ROD\_AUX(p,n,r)

MEMOIZZED\_CUT\_ROD\_AUX(p,n,r)

If r[n]>=0

Return r[n]

If n==0

q = 0

Else

q = max(p[0]+MEMOIZED\_CUT\_ROD\_AUX(p,n,r),p[n])

For i = 1 to n

q = max(q,

p[i]+MEMOIZED\_CUT\_ROD\_AUX(p,n-i,r)-ExtraFee)

r[n] = q

Return q

C++实现：

#include <QCoreApplication>

#include <iostream>

#define UNV -9999*//* *定义为负无穷*

#define EXFEE 2*//* *每次切割产生的额外费用*

using namespace std;

int Max(int a,int b)

{

return a>b?a:b;

}

int MEMOIZED\_CUT\_ROD\_AUX(int p[],int n,int r[])

{

static int q;

if (r[n]>=0)

{

return r[n];*//* *保存求过的值。*

}

if (n==0)

{

q=0;

}

else

{

*//若不切则不产生额外费用*

q = Max(p[n-1]+MEMOIZED\_CUT\_ROD\_AUX(p,0,r),p[0]+MEMOIZED\_CUT\_ROD\_AUX(p,n-1,r));

for (int i=1;i<n;i++)

{

*//若切割则每次产生额外费用*

q = Max(q, p[i]+MEMOIZED\_CUT\_ROD\_AUX(p,n-i-1,r)-EXFEE);

}

}

r[n]=q;

return q;

}

int MEMOIZED\_CUT\_ROD(int p[],int n)

{

int \*r=new int[n+1];

for (int i=0;i<n+1;i++)

{

r[i] = UNV;

}

int result = MEMOIZED\_CUT\_ROD\_AUX(p,n,r);

delete []r;

return result;

}

int main(int argc, char \*argv[])

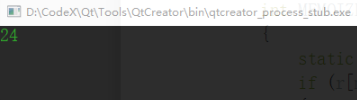
{

int arr[10] = {1, 5, 8, 9, 10, 17, 17, 20, 24, 30};

printf("%d",MEMOIZED\_CUT\_ROD(arr, 9));

return 0;

}



15.1-5设计一个O(n)时间的动态规划算法计算第n个斐波那契数(F0 = 0;F1 = 1;Fi = Fi-1 + Fi-2)。画出子问题图，图中有多少定点和边。

伪代码：

Fibonacci(array[], n)

If n >= 0

array[n] = 0

If n>=1

array[n] = 1

If n>=2

for i<-- 0 to n:

array[i] =array[i-1] + array[i-2]

C++代码描述：

int Fibonacci(int array[], int n)

{

if (n >= 0)

array[0] = 0;

if (n >= 1)

array[1] = 1;

if (n >= 2)

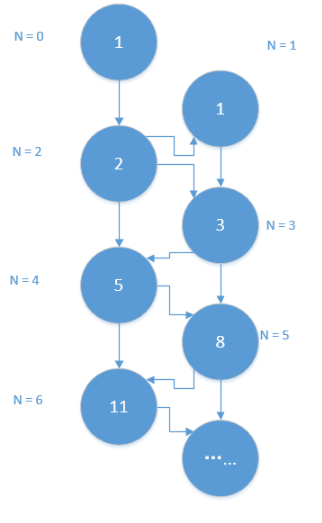
for (int i = 0;i <= n; i++) {

array[n] = Fibonacci(array, n-1) + Fibonacci(array, n-2);

}

return array[n];

}



由图易知，图中由n+1个顶点，2n+1条边。