# Project 6

**May 16,2022**

## Chapter1 INTRODUCTION

### 1.1 Background

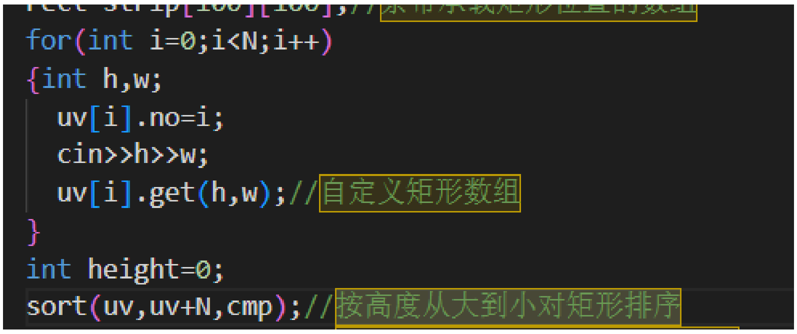
Texture packaging is the packaging of multiple rectangular textures into one large texture. The resulting texture must have a given width and minimum height.

### 1.2 Tasks specification

The best solution to this problem is a NP-hard problem, so we need to give an approximation algorithm to make the running time of the solution in polynomial time, and guarantee a certain approximation rate. We are going to design test samples, and different samples have squares of different heights and widths.

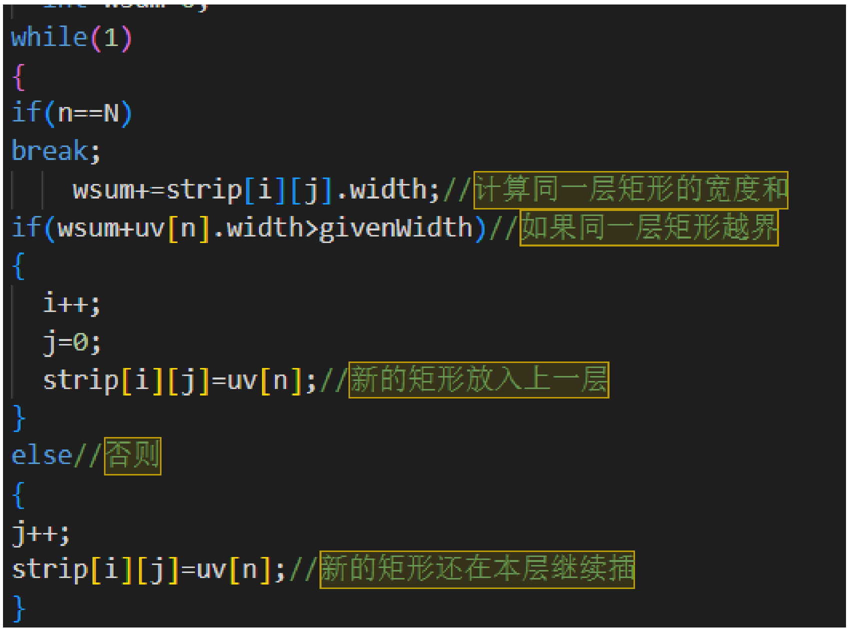
## Chapter2 ALGORITHM SPECIFICATION

First,we should sort rectangles in decreasing order by height.



```
rect strip[100][100];//
for(int i=0;i<N;i++)
{int h,w;
   uv[i].no=i;
   cin>>h>>w;
   uv[i].get(h,w);//自定义矩形数组
}
int height=0;
sort(uv,uv+N,cmp);//按高度从大到小对矩形排序
```

Second,we should check if the next rectangle can be packed into the current level.

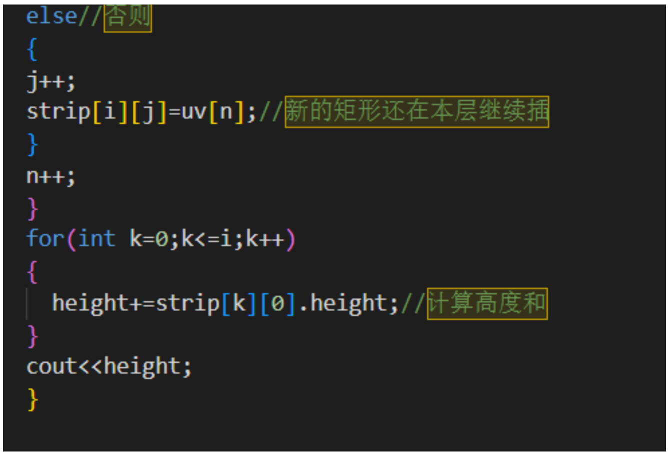


```
  int wsum=0;
while(1)
{
if(n==N)
break;
    wsum+=strip[i][j].width;//计算同一层矩形的宽度和
if(wsum+uv[n].width>givenWidth)//如果同一层矩形越界
{
  i++;
  j=0;
  strip[i][j]=uv[n];//新的矩形放入上一层
}
else//否则
{
j++;
strip[i][j]=uv[n];//新的矩形还在本层继续插
}
```

Third,we pack the rectangle.

a. Width of strip is exceeded: Place rectangle in a new level and justify left.

b. Width of strip not exceeded: Pack rectangle next to the previous rectangle



```
else//否则
{
j++;
strip[i][j]=uv[n];//新的矩形还在本层继续插
}
n++;
}
for(int k=0;k<=i;k++)
{
  height+=strip[k][0].height;//计算高度和
}
cout<<height;
}
```

## Chapter3 TESTING RESULTS

Test Cases

| Input | Goal | Output | State |
|---|---|---|---|
| 5<br>5<br>7 1<br>6 1<br>5 1<br>4 1<br>3 1 | Test the thickest sort of width | 7 | Pass |
| 6<br>10<br>3 6<br>3 4<br>3 4<br>3 6<br>3 5<br>3 5 | Test the case that each layer is paired and can fill the full width | 9 | Pass |

## Chapter4 COMPLEXITY ANALYSIS

### 4.1 Space complexity

Space complexity: an array of N rectangles with space complexity O(N) .

### 4.2 Time complexity

Time complexity: because of the need to call the sort function on the rectangular sorting according to height, so ordering module time complexity is O (NlogN), cyclic selection module are traversed each rectangle and through the calculate and determine the should be inserted into the layer, so choose module time complexity is O (N), the total time complexity O (NlogN).

## Chapter5 CODE APPENDIX

```
#include<iostream>
#include<algorithm>

using namespace std;
class rect{//新建矩形类
  public:
int no;
int height;
int width;
rect(){};
rect(int h,int w)
{
  height=h;
  width=w;
}
void get(int h,int w)//获取矩形的高度和宽度
{cout<<"yes";
  height=h;
  width=w;
}
};
bool cmp(rect x,rect y){
    return x.height>y.height;//用以比较矩形高度的函数
}
int main()
{
int givenWidth;//给定的条带宽度
int N;//有几个矩形
cin>>N;
cin>>givenWidth;
rect uv[N];//存储矩形的数组
rect strip[100][100];//条带承载矩形位置的数组
for(int i=0;i<N;i++)
{int h,w;
  uv[i].no=i;
  cin>>h>>w;
  uv[i].get(h,w);//自定义矩形数组
}
int height=0;
sort(uv,uv+N,cmp);//按高度从大到小对矩形排序
strip[0][0]=uv[0];//高度最高的矩形放入条带的左下
int i=0;
int j=0;
int n=1;
  int wsum=0;
while(1)
{
if(n==N)
break;
    wsum+=strip[i][j].width;//计算同一层矩形的宽度和
if(wsum+uv[n].width>givenWidth)//如果同一层矩形越界
{
  i++;
  j=0;
  strip[i][j]=uv[n];//新的矩形放入上一层
}
else//否则
{
j++;
strip[i][j]=uv[n];//新的矩形还在本层继续插
}
n++;
}
for(int k=0;k<=i;k++)
{
  height+=strip[k][0].height;//计算高度和
}
cout<<height;
}
```