中山大学数据科学与计算机学院本科生实验报告

（2016 学年春季学期）

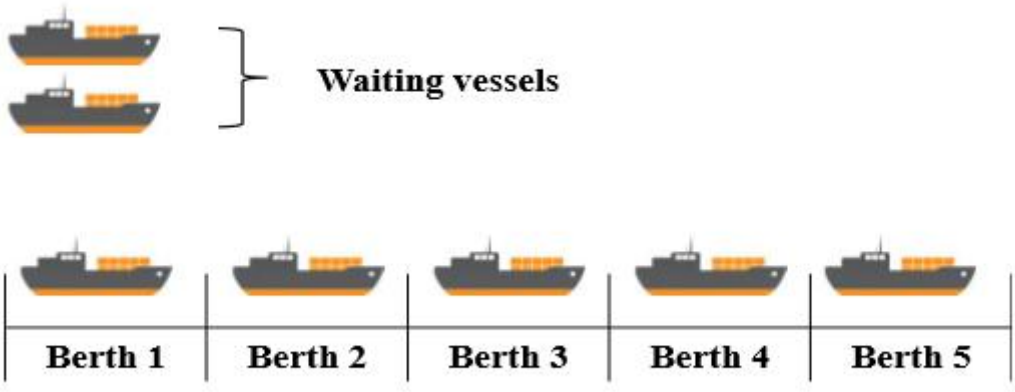
课程名称：Algorithm design　　任课教师：张子臻

年级	2015 级	专业（方向）	软件工程（移动信息工程）
学号	15352461	姓名	宗嘉希
电话	18022724490	Email	zongjx@mail2.sysu.edu.cn
开始日期	2017.05.26	完成日期	2017.06.02

1.实验题目

Breth Allocation Problem

In a container terminal, such as the Hong Kong Container Terminal, the bottleneck of the traffic is often at the quay. Therefore, the terminal operator has to allocate a limited number of berths of the quay to vessels in an efficient way.



Consider a container terminal of n berths and m vessels arrived, where each vessel requires one or more berths to load and unload containers. Vessel i (for i=1,2,…,m) arrives at time ai with its service time t_i hours, and it occupies b_i berths. For each vessel i=1,2,…m, the terminal manager needs to decide on the berths, as well on the starting time si of berthing

for it. It must be satisfied that no two vessels are allowed to occupy the same berth simultaneously, i.e., for any two different vessels i and j, if bi=bj, then either $s_i+t_i <= s_j$or $s_j+t_j <= s_j$ must be satisfied. Given the limited time horizon, your task is to help the manager to minimize the unassigned vessels, the total waiting time and the last departure time of all the vessels.

2.实验目的

通过编写算法，解决 BAP 问题，使得船舶停靠方案最优。（最优方案：有三个评判标准：1、未停靠的船舶的数量　　2、船舶等待停靠的总用时　　3、最后一艘船离开的时间　　三个评判标准各有一个权值，每个权值分别与各项相乘求和，得到的结果为总花费，最优方案要求总花费最少。）

3.程序设计

首先我们可以分析一下，这一个问题应该是一个 NP 完全问题了。确实目前没有有一个可以完美解决这一个问题的算法，而我也没有想到什么非常有效率的、可行的方法。我所使用的算法是最随便、最简单、最无脑的贪心算法。主要的思路是先把所有的船按照到达的时间升序排序，然后从左到右，从上到下，按顺序把船放置在第一个可以放置的地方。这一个思路是非常简单的，但是在大多数的情况下是不可能达到最优解的。要是想获取最优解，但是又想要简单暴力，那就只有搜索了。搜索可以得到所有有可能的情况，然后通过计算每一种情况的花费，取最少花费的那一个结果。但是很明显，使用搜索的话在一些简单的样例中可以比较快地得到结果，然而在一些很复杂的情况中，完全搜索的话要花费大量的时间（以小时为单位），因此虽然搜索可以得到最优解，但是由于运算时间不太合理，我并不会采用这种方法（其实是懒得写，不过总不可能用搜索去验收吧）。除了这两种方法以外，我并没有想到有什么比较好的方法。因此我决定在基于简单的贪心算法，针对一些特殊的样例做一些改进。针对了样例 2，发现在贪心的过程中会把一些服务时间较长的船只放在中间的位置，导致有一些比较长，服务时间也长得船只没有得到应有的合理的位置去放置。所以我做了一个算法去尽量避免这一种情况的发生。我认为基

本上不可能再多做出几个优化的方案了，因为当我对一种特殊情况进行优化的时候，很有可能就会影响到其他的问题的解决，但是由于能力有限，我也只能做到这种地步了。

首先说明一下，我是使用类来定义船只和港口的，相关的一些功能函数都写在了类里面：

船类的声明：

```
class vessel{
    public:
        vessel();
        ~vessel();
        void set_data(int arrival_time_,int service_time_,int breths_occupied_);
        int get_arrival_time();
        int get_service_time();
        int get_breths_occupied();
        int get_assign_time();
        void set_assign_time(int arrival_time);
        int get_waiting_time();
        vessel& operator=(const vessel& v);
    private:
        int arrival_time;
        int service_time;
        int breths_occupied;
        int assign_time;
};
```

港口类的声明

```
class breth{
    public:
        breth(int total_time_,int total_breth_);
        ~breth();
        void set_total_time(int total_time_);
        void set_total_breth(int total_breth_);
        void op1();
        void op2();
        void occupy(int time,int breth,int index,int xs,int ys);
        bool check(int time,int breth,int xs,int ys);
        void arrive(vessel &vessels,int index);
        int cal_last_departure_time();
        int *maxtime;
    private:
        int total_time;
        int total_breth;
        int** location;
};
```

在完成对两个类的声明以后，接下来就是要实现类里面的成员函数了，当然，本次的算法包含在港口类的成员函数里面。

对于成员函数的实现，我就不在此一一截图展示了，在此稍作说明。船的类里面无非就是存放那几个关键的参数：到达时间、服务时间、占用港口的数量。为了方便标记还多设置了一个放置的时间。

港口类相对来说就要复杂一点了，除了基本的参数（港口的总大小、服务总时间），还有存放了一个港口的模型、对港口的操作（填充位置，计算耗时等等），还有检查、放置船只等函数。最重要的贪心算法也写在里面了。

程序的界面只是小问题，我本来的思路是：开始时先输入三个权值（由于是要求最优解，所以肯定是要按照权值来找出最小花费的解，权值不一样，摆法也不一样），输入完权值以后，根据权值搜索出最优解。然而放弃了搜索以后，我使用了贪心（这样的话解就只有一个了，虽然不是最优）。因此我把权值的输入放在最前。按照题目给出的要求去输入参数，输入完参数以后，得到结果、计算出总花费。还可以输出船只摆放的示意图（简陋版）。

以下是我的实现摆放的算法部分（贪心算法加上针对某些特殊情况的改进，请原谅我的菜）：

```
144 void breth::arrive(vessel &vessels,int index){
145     for(int i=vessels.get_arrival_time();i<total_time;i++){
146         for(int k=0;k<total_breth;k++){
147             if(check(vessels.get_service_time(),vessels.get_breths_occupied(),k,i)){
148                 if(vessels.get_service_time()+i>maxtime[k]&&k!=0){
149                     if(k+vessels.get_breths_occupied()!=total_breth){
150                         if(k+vessels.get_breths_occupied()<total_breth){
151                             if(location[k+vessels.get_breths_occupied()][i]!=-1){
152                                 occupy(vessels.get_service_time(),vessels.get_breths_occupied(),index,k,i);
153                                 ans[2*(index-1)]=k;
154                                 ans[2*(index-1)+1]=i;
155                                 vessels.set_assign_time(i);
156                                 for(int p=k;p<total_breth;p++){
157                                     if(maxtime[p]<i+vessels.get_service_time())
158                                     maxtime[p]=i+vessels.get_service_time();
159                                 }
160                                 return;
161                             }
162                             else{
163                                 continue;
164                             }
165                         }
166                         else continue;
167                     }
168                     else{
169                         occupy(vessels.get_service_time(),vessels.get_breths_occupied(),index,k,i);
170                         ans[2*(index-1)]=k;
171                         ans[2*(index-1)+1]=i;
172                         vessels.set_assign_time(i);
173                         for(int p=k;p<total_breth;p++){
174                             if(maxtime[p]<i+vessels.get_service_time())
175                             maxtime[p]=i+vessels.get_service_time();
176                         }
177                         return;
178                     }
179                 }
180                 else{
181                     occupy(vessels.get_service_time(),vessels.get_breths_occupied(),index,k,i);
182                     ans[2*(index-1)]=k;
183                     ans[2*(index-1)+1]=i;
184                     vessels.set_assign_time(i);
185                     for(int p=k;p<total_breth;p++){
186                         if(maxtime[p]<i+vessels.get_service_time())
187                         maxtime[p]=i+vessels.get_service_time();
188                     }
189                     return;
190                 }
191
192             }
193         }
194     }
195 }
```

贪心的思路就不用详细解释了，其实就是遍历所有的船，然后从左到右、从上到下找到第一个可以放置该船的位置把该船放下。基于贪心的思路，我又做了了一些小的修改。由于贪心算法对于某些情况来说是不合理的，因此要对解的某些地方做一些局部的调整，因此对一种特殊的情况做了一些局部的调整。首先找到一艘船停放的位置，然后再看该船结束服务的时候是否有停靠在它之前的港口而结束服务时间又比它小的船，如果存在的话就尽量这一艘船停靠的港口往后移（不影响等待时间）。这样的话就可以节省出一段比较长的连续的位置给一些需要较长港口的船停留。使用一个数组来记录从 0 号港口到目前船只停留位置的港口中最长的服务时间，这样就可以很轻松地找出是否有符合规则的情况。通常情况下只有停留在 0 号港口的船只可以影响到这个数组。

然而，很显然只针对了某一种情况做出了局部的调整，有很大的几率会影响到别的情况，有时候甚至做出来的结果比贪心还差，因此，为了避免出现得到更差的解，在使用贪心算法去做一次，然后比较两种算法哪一种可以得出更优的解，再去较优的那个解（太菜只能做到这个地步）。

以下是贪心的算法：

```
199 void breth::arrive2(vessel &vessels,int index){
200     for(int i=vessels.get_arrival_time();i<total_time;i++){
201         for(int k=0;k<total_breth;k++){
202             if(check(vessels.get_service_time(),vessels.get_breths_occupied(),k,i)){
203                 occupy(vessels.get_service_time(),vessels.get_breths_occupied(),index,k,i);
204                 ans2[2*(index-1)]=k;
205                 ans2[2*(index-1)+1]=i;
206                 vessels.set_assign_time(i);
207                 for(int p=k;p<total_breth;p++){
208                     if(maxtime[p]<i+vessels.get_service_time())
209                     maxtime[p]=i+vessels.get_service_time();
210                 }
211                 return;
212             }
213         }
214     }
215 }
```

做到这种地步我已经尽力了。最后我发现了有一个地方还可以做一点小优化（第十一个样例），因此增加了第三个算法，但是第三个算法跟改进算法的区别即只有一个判断，这里不详细说明，实际上只是改变了起始的位置而已。

在 DDL 之前，我发现我这样子做实在是太水了，所以又重新写了一个搜索的算法：先把船全排列，然后再使用贪心算法去搜索最优解，在我提交的结果中，使用全排列贪心搜索的结果应该都是最优的，

但是我只在船只比较少的情况下使用这种算法（大概在 12 艘船以内），因为耗时实在是太夸张了，我也不懂得如何去进行一下优化。后来我觉得起码要为一个样例都输出一下结果，因此我就设置了一个值 i，每产生一次结果，都与目前记录的最优解作比较，如果该解更优，就替换最优解，并把 i 置 0；如果不能替换最优解，那么 i 就加 1，如果 i 达到了某一个值 n，也就是说经历过了 n 种情况没有更新最优解，那么就取目前最优解输出。（这只是为了中断要等很长时间的程序的运行）。经过一定的改进，我让他没有更新最优解的时候再换一个排列（更换全排列的第一位），继续以上搜索操作，直到全排列完成（其实这个过程中有很多没有排列出来）。关键代码如下：

```
void solve_pre(breth &new_breth,vessel *vessels,int a[]){        //放船
    memset(ans2,-1,sizeof(ans2));
    for(int i=1;i<=m;i++){
        int k=a[i];

        new_breth.arrive(vessels[k],k);
    }
    int count_of_not_arrive=0,count_of_time=0,the_last_departure_time=0;
    for(int i=1;i<=m;i++){
        if(vessels[i].get_waiting_time()==-1)
        count_of_not_arrive++;
        else{
            count_of_time+=vessels[i].get_waiting_time();
        }
    }
    the_last_departure_time=new_breth.cal_last_departure_time();
    int punish=0;
    punish=count_of_not_arrive*weight_of_unassigned_vessels+count_of_time*weight_of_total_waiting_time+the_last_departure_time
    if(punish<new_breth.best_punish){
        new_breth.change(punish,count_of_time,count_of_not_arrive,the_last_departure_time);
        interrupt=0;
    }
    else interrupt++;
}
void f1(int visited[],int y,breth &new_breth,vessel *vessels)//递归求全排列
{
    int i;
    if(y==m+1)
    {
        new_breth.op1();
        for(int i=1;i<=m;i++)
        vessels[i].erase_mark();
        solve_pre(new_breth,vessels,a);
    }
    else
    {
        for(i=1;i<=m;i++)
        {
            if(visited[i]==0)
            {
                a[y]=i;
                visited[i]=1;
                if(interrupt>=400000) return;     ///中断搜索
                f1(visited,y+1,new_breth,vessels);
                visited[i]=0;
            }
        }
    }
}
```

```
void solve(breth &new_breth,vessel *vessels)   //求解（求全排列）
{
    int l;
    int temp[1001];
        for(l=1;l<=m;l++)
        {
            memset(temp,0,sizeof(temp));
            memset(a,0,sizeof(a));
            a[1]=l;
            temp[l]=1;
            f1(temp,2,new_breth,vessels);
            cout<<l<<endl;
            if(interrupt>=400000) {
                interrupt=0;
                continue;
            }
        }
}
```

没有做到随机性，是有点可惜，不然就可以找到更多的解了。

4.程序运行与测试

程序运行的界面如下：

```
                                BAP
please input the weight of the unassigned vessels:
```

首先我要输入三个权值，使得输入所有的数据以后可以判断出哪一种算法得到的答案的总花费最少，随后再输入港口的大小，船只的数量，船只的相关数据等等：

```
                                BAP
please input the weight of the unassigned vessels: 100
please input the weight of total waiting time: 2
please input the weight oflast departure time: 1
please input the total time of the time: 6
please input the total length of the breth: 6
please input the number of vessels: 5
please input the 1st vessel's data (arrival time , service time , breths occupued):
 0 3 5
please input the 2nd vessel's data (arrival time , service time , breths occupued):
 1 1 2
please input the 3rd vessel's data (arrival time , service time , breths occupued):
 2 1 2
please input the 4th vessel's data (arrival time , service time , breths occupued):
 2 1 5
please input the 5th vessel's data (arrival time , service time , breths occupued):
 2 2 4
```

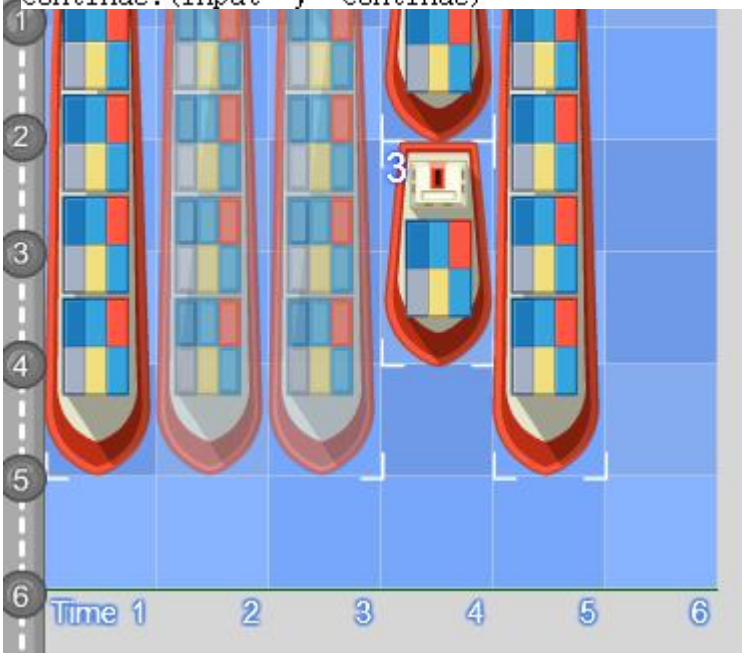
输入完以后，就会得到输出的结果（包括坐标形式的解和三个判断标准的值），也可以输出简陋的船只停

放示意图：

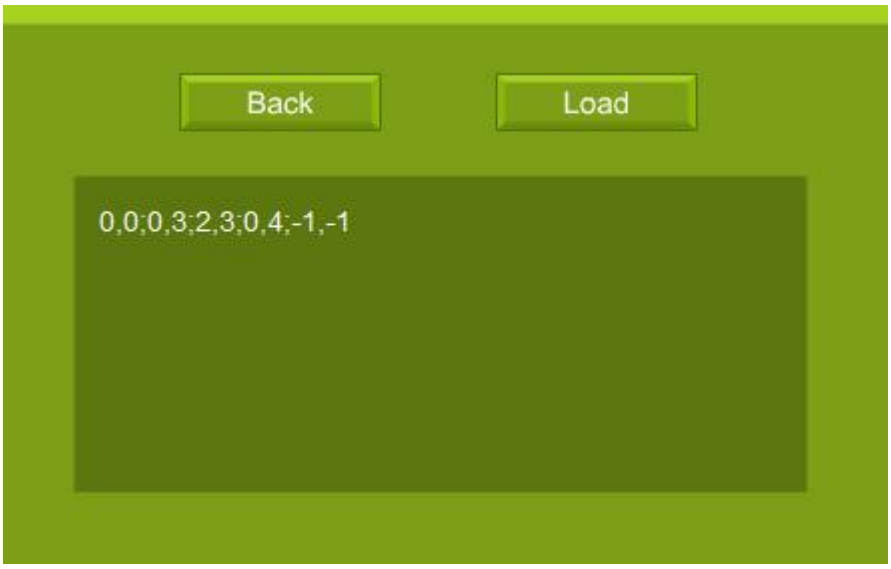
```
Answer:  0,0;0,3;2,3;0,4;-1,-1
The number of unassigned vessels is: 1
The total waiting time is: 5
The last departure time is: 5
The total cost is: 115

input 'y' to show the breth or input other to skip y
1 1 1 2 4 0
1 1 1 2 4 0
1 1 1 3 4 0
1 1 1 4 4 0
1 1 1 0 4 0
0 0 0 0 0 0

continue?(input 'y' continue)
```



以上是样例一的解（贪心），解出来跟网站上贪心的结果一样：



样例二的解与样例一的稍有不同，因为样例二的解使用改进过的算法会得到更好的结果：

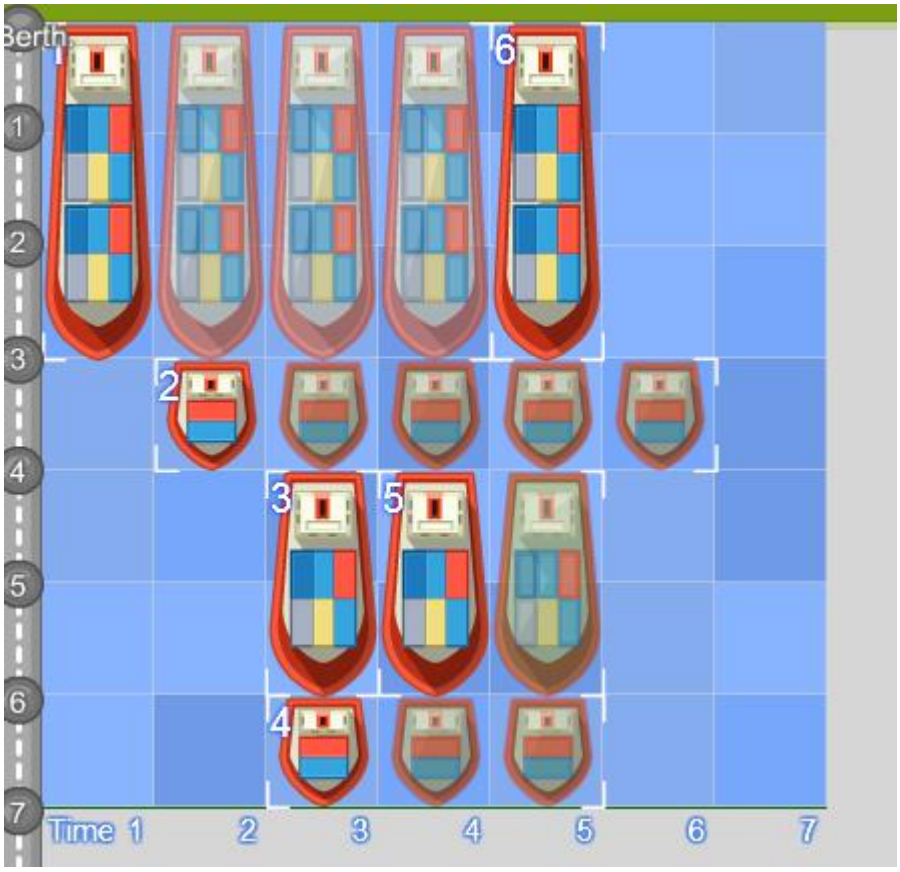
```
                                    BAP
please input the weight of the unassigned vessels: 100
please input the weight of total waiting time: 2
please input the weight oflast departure time: 1
please input the total time of the time: 7
please input the total length of the breth: 7
please input the number of vessels: 7
please input the 1st vessel's data (arrival time , service time , breths occupue
d): 0 4 3
please input the 2nd vessel's data (arrival time , service time , breths occupue
d): 1 5 1
please input the 3rd vessel's data (arrival time , service time , breths occupue
d): 2 1 2
please input the 4th vessel's data (arrival time , service time , breths occupue
d): 2 3 1
please input the 5th vessel's data (arrival time , service time , breths occupue
d): 3 2 2
please input the 6th vessel's data (arrival time , service time , breths occupue
d): 4 1 3
please input the 7th vessel's data (arrival time , service time , breths occupue
d): 5 2 5

Answer:  0,0;6,1;3,2;5,2;3,3;0,4;0,5
The number of unassigned vessels is: 0
The total waiting time is: 0
The last departure time is: 7
The total cost is: 7

input 'y' to show the breth or input other to skip y
1 1 1 1 6 7 7
1 1 1 1 6 7 7
1 1 1 1 6 7 7
0 0 3 5 5 7 7
0 0 3 5 5 7 7
0 0 4 4 4 0 0
0 2 2 2 2 2 0

continue?(input 'y' continue) _
```

可以看见，使用改进过的算法是可以把所有的船的放进去的，而只使用贪心的话，结果就会像网站上的一样，甚至还没摆满：



可见已经做出了一些优化。

样例三：

```
                                    BAP
please input the weight of the unassigned vessels: 100
please input the weight of total waiting time: 2
please input the weight oflast departure time: 1
please input the total time of the time: 17
please input the total length of the breth: 4
please input the number of vessels: 7
please input the 1st vessel's data (arrival time , service time , breths occupue
d): 0 1 1
please input the 2nd vessel's data (arrival time , service time , breths occupue
d): 1 3 3
please input the 3rd vessel's data (arrival time , service time , breths occupue
d): 2 4 3
please input the 4th vessel's data (arrival time , service time , breths occupue
d): 2 8 2
please input the 5th vessel's data (arrival time , service time , breths occupue
d): 3 7 1
please input the 6th vessel's data (arrival time , service time , breths occupue
d): 3 2 1
please input the 7th vessel's data (arrival time , service time , breths occupue
d): 4 7 2

Answer:  0,0;0,1;0,4;0,8;3,3;2,8;2,10
The number of unassigned vessels is: 0
The total waiting time is: 19
The last departure time is: 17
The total cost is: 55

input 'y' to show the breth or input other to skip y
1 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 0
0 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 0
0 2 2 2 3 3 3 3 6 6 7 7 7 7 7 7 7
0 0 0 5 5 5 5 5 5 5 7 7 7 7 7 7 7
```

样例四：

```
please input the 1st vessel's data (arrival time , service time , breths occupue
d): 0 5 2
please input the 2nd vessel's data (arrival time , service time , breths occupue
d): 1 2 3
please input the 3rd vessel's data (arrival time , service time , breths occupue
d): 1 2 3
please input the 4th vessel's data (arrival time , service time , breths occupue
d): 2 2 4
please input the 5th vessel's data (arrival time , service time , breths occupue
d): 3 2 2
please input the 6th vessel's data (arrival time , service time , breths occupue
d): 4 5 4
please input the 7th vessel's data (arrival time , service time , breths occupue
d): 5 4 2
please input the 8th vessel's data (arrival time , service time , breths occupue
d): 6 3 5
please input the 9th vessel's data (arrival time , service time , breths occupue
d): 8 1 4
please input the 10th vessel's data (arrival time , service time , breths occupu
ed): 8 1 3

Answer:  0,0;2,1;2,3;0,5;0,7;0,9;0,14;-1,-1;0,18;2,8
The number of unassigned vessels is: 1
The total waiting time is: 33
The last departure time is: 19
The total cost is: 185

input 'y' to show the breth or input other to skip y
1 1 1 1 1 4 4 5 5 6 6 6 6 6 6 7 7 7 7 9 0
1 1 1 1 1 4 4 5 5 6 6 6 6 6 6 7 7 7 7 9 0
0 2 2 3 3 4 4 0 10 6 6 6 6 6 0 0 0 0 9 0
0 2 2 3 3 4 4 0 10 6 6 6 6 6 0 0 0 0 9 0
0 2 2 3 3 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0

continue?(input 'y' continue)
```

样例五：

```
Answer:  0,0;1,0;5,0;6,0;1,1;2,1;0,2;3,2;6,3;7,3;11,3;0,5;7,4;-1,-1;0,6;4,6;6,7
The number of unassigned vessels is: 1
The total waiting time is: 4
The last departure time is: 9
The total cost is: 117

input 'y' to show the breth or input other to skip y
1 1 7 7 0 12 15 15 15
2 5 7 7 0 12 15 15 15
2 6 7 7 0 12 15 15 15
2 6 8 8 8 12 15 15 15
2 0 8 8 8 12 16 16 0
3 3 8 8 8 0 16 16 0
4 4 4 9 9 9 9 17 0
4 4 4 10 13 13 13 13 0
4 4 4 10 13 13 13 13 0
4 4 4 10 13 13 13 13 0
0 0 0 10 0 0 0 0 0
0 0 0 11 11 11 11 0 0
```

样例六：

```
Answer:  0,0;2,1;6,1;10,1;2,3;4,3;5,6;5,7;0,8;-1,-1
The number of unassigned vessels is: 1
The total waiting time is: 7
The last departure time is: 10
The total cost is: 124

input 'y' to show the breth or input other to skip y
1 1 1 1 1 0 0 0 9 9
1 1 1 1 1 0 0 0 9 9
0 2 2 5 5 5 5 5 9 9
0 2 2 5 5 5 5 5 9 9
0 2 2 6 6 6 6 6 9 9
0 2 2 0 0 0 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 4 4 4 4 4 0 0 0 0
0 4 4 4 4 4 0 0 0 0
```

样例七：

```
Answer:  0,0;10,0;6,0;0,1;0,2;4,4;9,5;4,5;4,2;0,6;2,6;2,8;4,9;-1,-1
The number of unassigned vessels is: 1
The total waiting time is: 25
The last departure time is: 10
The total cost is: 160

input 'y' to show the breth or input other to skip y
1 4 5 5 5 5 10 10 10 10
1 4 5 5 5 5 10 10 10 10
0 4 5 5 5 5 11 11 12 12
0 4 5 5 5 5 11 11 12 12
0 4 9 9 6 8 8 8 8 13
0 0 9 9 6 8 8 8 8 13
3 3 3 3 6 8 8 8 8 13
3 3 3 3 6 8 8 8 8 13
3 3 3 3 0 8 8 8 8 0
3 3 3 3 0 7 7 7 7 7
2 2 2 2 2 7 7 7 7 7
2 2 2 2 2 7 7 7 7 7
```

样例八：

```
Answer:  0,0;5,2;9,2;5,6;0,7;3,7;0,9;6,11;3,11;0,12;4,11;-1,-1
The number of unassigned vessels is: 1
The total waiting time is: 32
The last departure time is: 15
The total cost is: 179

input 'y' to show the breth or input other to skip y
1 1 1 1 1 1 1 5 5 7 7 7 10 10 10 0
1 1 1 1 1 1 1 5 5 7 7 7 10 10 10 0
1 1 1 1 1 1 1 5 5 7 7 7 10 10 10 0
1 1 1 1 1 1 1 6 6 6 6 9 10 10 10 0
1 1 1 1 1 1 1 6 6 6 6 11 11 11 0 0
0 0 2 2 2 2 4 4 4 4 4 0 0 0 0 0
0 0 2 2 2 2 4 4 4 4 4 8 8 0 0 0
0 0 2 2 2 2 4 4 4 4 4 8 8 0 0 0
0 0 2 2 2 2 4 4 4 4 4 8 8 0 0 0
0 0 3 3 3 0 4 4 4 4 4 8 8 0 0 0
```

样例九：

```
Answer:  0,0;4,0;0,2;5,2;0,6;0,7;0,11;0,14;4,7;0,15;3,6;4,9;-1,-1;5,11;4,14
The number of unassigned vessels is: 1
The total waiting time is: 47
The last departure time is: 19
The total cost is: 213

input 'y' to show the breth or input other to skip y
1 0 3 3 3 3 5 6 6 6 6 7 7 7 8 10 10 10 10 0
1 0 3 3 3 3 5 6 6 6 6 7 7 7 8 10 10 10 10 0
1 0 3 3 3 3 5 6 6 6 6 7 7 7 8 10 10 10 10 0
1 0 3 3 3 3 11 6 6 6 6 7 7 7 0 10 10 10 10 0
2 2 2 2 2 0 0 9 9 12 12 7 7 7 15 15 15 15 0 0
0 0 4 4 4 4 4 9 9 12 12 14 14 14 14 14 0 0 0 0
```

样例十：

```
Answer:  0,0;8,0;4,0;0,1;0,5;2,1;0,7;5,5;5,10;0,11;4,13;8,4;0,14;3,14;5,17;4,11;
0,17;2,14;2,16
The number of unassigned vessels is: 0
The total waiting time is: 101
The last departure time is: 20
The total cost is: 222

input 'y' to show the breth or input other to skip y
1 4 4 4 4 5 5 7 7 7 7 10 10 10 13 13 13 17 0 0
0 4 4 4 4 5 5 7 7 7 7 10 10 10 13 13 13 17 0 0
0 6 6 6 6 5 5 7 7 7 7 10 10 10 18 18 19 17 0 0
0 6 6 6 6 5 5 7 7 7 7 10 10 10 14 14 14 14 0 0
3 3 3 3 3 5 5 7 7 7 7 16 16 11 11 11 11 0 0 0
3 3 3 3 3 8 8 8 8 8 9 9 9 11 11 11 11 15 15 15
3 3 3 3 3 8 8 8 8 8 9 9 9 11 11 11 11 15 15 15
3 3 3 3 3 8 8 8 8 8 9 9 9 11 11 11 11 15 15 15
2 2 2 0 12 8 8 8 8 8 9 9 9 11 11 11 11 15 15 15
```

样例十一：

```
Answer:  0,0;5,0;0,5;4,5;0,10;5,7;6,2;0,15;0,18;5,12;6,3;0,21;7,3;7,12;5,4;4,7;-1,-1;
-1,-1;5,18;4,9
The number of unassigned vessels is: 2
The total waiting time is: 98
The last departure time is: 24
The total cost is: 420

input 'y' to show the breth or input other to skip y
1 1 1 1 1 3 3 3 3 3 5 5 5 5 5 8 8 8 9 9 9 12 12 12
1 1 1 1 1 3 3 3 3 3 5 5 5 5 5 8 8 8 9 9 9 12 12 12
1 1 1 1 1 3 3 3 3 3 5 5 5 5 5 8 8 8 9 9 9 12 12 12
1 1 1 1 1 3 3 3 3 3 5 5 5 5 5 8 8 8 9 9 9 12 12 12
1 1 1 1 1 4 4 16 16 20 5 5 5 5 5 0 0 0 9 9 9 12 12 12
2 2 2 2 15 4 4 6 6 6 6 6 10 10 10 10 10 10 19 0 0 0 0 0
0 0 7 11 15 4 4 6 6 6 6 6 10 10 10 10 10 10 19 0 0 0 0 0
0 0 7 13 0 4 4 6 6 6 6 6 14 14 14 14 14 0 19 0 0 0 0 0
```

样例十二：

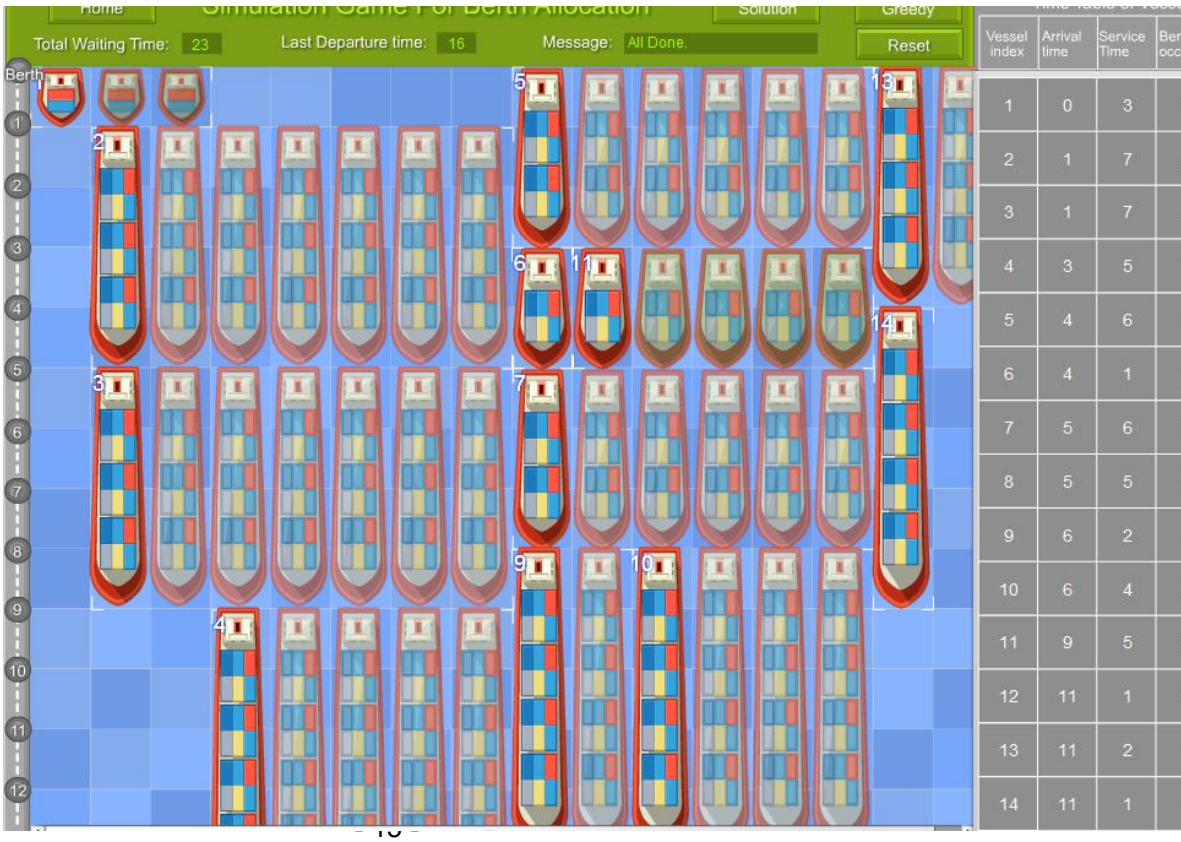
```
Answer:  0,0;1,0;2,0;5,2;6,2;0,6;5,8;5,9;10,5;0,11;1,11;4,13;7,13;8,13;10,13;8,15;
8,17;-1,-1;4,20;0,23;5,23;11,17;-1,-1;9,23
The number of unassigned vessels is: 2
The total waiting time is: 61
The last departure time is: 29
The total cost is: 351

input 'y' to show the breth or input other to skip y
1 1 1 0 0 0 6 6 6 6 6 10 10 10 10 10 10 10 10 0 0 0 0 20 20 20 0 0 0 0
2 2 2 2 2 2 6 6 6 6 6 11 11 11 11 11 11 11 11 11 11 11 11 20 20 20 0 0 0 0
3 3 3 3 3 0 6 6 6 6 6 11 11 11 11 11 11 11 11 11 11 11 11 20 20 20 0 0 0 0
3 3 3 3 3 0 6 6 6 6 6 11 11 11 11 11 11 11 11 11 11 11 11 20 20 20 0 0 0 0
3 3 3 3 3 0 6 6 6 6 6 0 0 12 12 12 12 12 12 12 19 0 0 20 20 20 0 0 0 0
0 0 4 4 4 4 0 0 7 8 8 8 8 12 12 12 12 12 12 12 19 0 0 21 21 21 21 21 21 0
0 0 5 5 5 5 5 5 7 8 8 8 8 12 12 12 12 12 12 12 19 0 0 21 21 21 21 21 21 0
0 0 5 5 5 5 5 5 7 8 8 8 8 13 13 13 13 13 13 13 13 0 21 21 21 21 21 21 0
0 0 5 5 5 5 5 5 7 8 8 8 8 14 14 16 16 17 17 17 17 17 17 21 21 21 21 21 21 0
0 0 5 5 5 5 5 5 0 8 8 8 8 14 14 16 16 17 17 17 17 17 17 24 24 24 24 0 0 0
0 0 0 0 0 9 9 9 9 9 9 9 9 15 15 15 15 15 15 15 15 15 15 0 0 0 0 0 0 0
0 0 0 0 0 9 9 9 9 9 9 9 9 0 0 0 0 22 22 22 22 22 0 0 0 0 0 0 0 0
```

脱离样例，我自己也随机创建了几组 problem，相关的数据我写在了附件的 excel 文档里面，下面来看一下结果：

新样例 1：

这是贪心的结果：

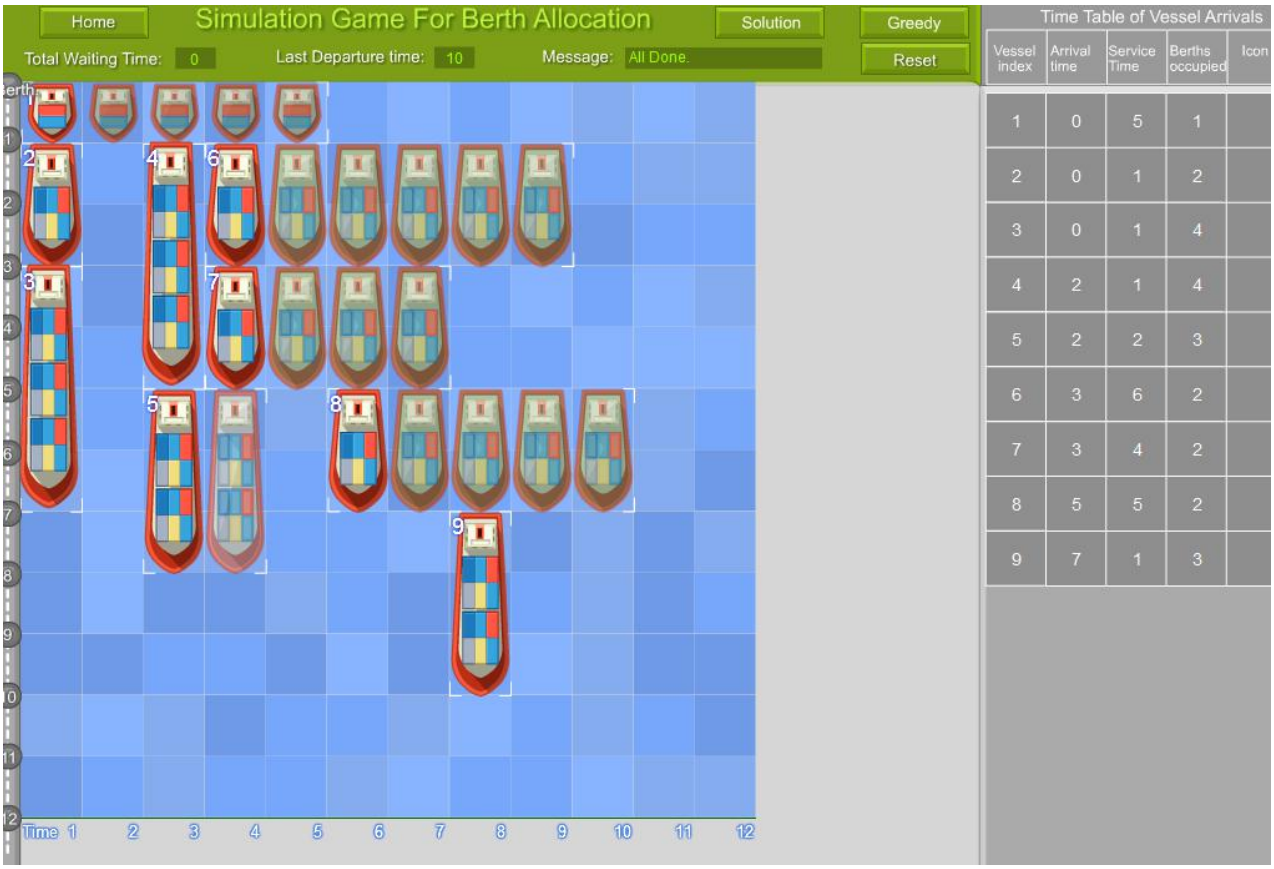


程序运行结果：

```
Answer:  0,0;12,1;8,1;0,3;5,4;0,8;2,8;8,8;9,8;9,10;0,9;5,11;5,13;(
The number of unassigned vessels is: 0
The total waiting time is: 21
The last departure time is: 15
The total cost is: 57

input 'y' to show the breth or input other to skip y
1 1 1 4 4 4 4 4 6 11 11 11 11 11 14
0 0 0 4 4 4 4 4 6 11 11 11 11 11 14
0 0 0 4 4 4 4 4 7 7 7 7 7 7 14
0 0 0 4 4 4 4 4 7 7 7 7 7 7 14
0 0 0 4 4 4 4 4 7 7 7 7 7 7 14
0 0 0 0 5 5 5 5 5 5 0 12 0 13 13
0 0 0 0 5 5 5 5 5 5 0 12 0 13 13
0 0 0 0 5 5 5 5 5 5 0 0 0 13 13
0 3 3 3 3 3 3 3 8 8 8 8 8 13 13
0 3 3 3 3 3 3 3 9 9 10 10 10 10 0
0 3 3 3 3 3 3 3 9 9 10 10 10 10 0
0 3 3 3 3 3 3 3 9 9 10 10 10 10 0
0 2 2 2 2 2 2 2 9 9 10 10 10 10 0
0 2 2 2 2 2 2 2 9 9 10 10 10 10 0
0 2 2 2 2 2 2 2 0 0 0 0 0 0 0
0 2 2 2 2 2 2 2 0 0 0 0 0 0 0
```

新样例 2：



程序运行结果：

```
Answer:  0,0;1,0;3,0;1,2;5,2;3,3;1,3;10,5;0,7
The number of unassigned vessels is: 0
The total waiting time is: 0
The last departure time is: 10
The total cost is: 10

input 'y' to show the breth or input other to skip y
1 1 1 1 1 0 0 9 0 0 0 0
2 0 4 7 7 7 7 9 0 0 0 0
2 0 4 7 7 7 7 9 0 0 0 0
3 0 4 6 6 6 6 6 6 0 0 0
3 0 4 6 6 6 6 6 6 0 0 0
3 0 5 5 0 0 0 0 0 0 0 0
3 0 5 5 0 0 0 0 0 0 0 0
0 0 5 5 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 8 8 8 8 8 0 0
0 0 0 0 0 8 8 8 8 8 0 0
```

注意：我写了两个程序，贪心是在 project1.cpp

全排列是在 project_2.cpp 中

使用新的算法后，有几个样例得到了明显的提升（与贪心对比起来，速度慢很多，但是大多数的解都得到了显著的提升）：

样例 1：

```
Answer:  0,0;0,3;0,4;0,5;2,3
The number of unassigned vessels is: 0
The total waiting time is: 8
The last departure time is: 6
The total cost is: 22

input 'y' to show the breth or input other to skip y
1 1 1 2 3 4
1 1 1 2 3 4
1 1 1 5 5 4
1 1 1 5 5 4
1 1 1 5 5 4
0 0 0 5 5 0
```

样例 4：

```
Answer:  0,0;2,1;2,3;0,10;0,5;0,15;3,5;0,12;0,9;0,8
The number of unassigned vessels is: 0
The total waiting time is: 30
The last departure time is: 20
The total cost is: 80

input 'y' to show the breth or input other to skip y
1 1 1 1 1 5 5 0 10 9 4 4 8 8 8 6 6 6 6 6
1 1 1 1 1 5 5 0 10 9 4 4 8 8 8 6 6 6 6 6
0 2 2 3 3 0 0 0 10 9 4 4 8 8 8 6 6 6 6 6
0 2 2 3 3 7 7 7 7 9 4 4 8 8 8 6 6 6 6 6
0 2 2 3 3 7 7 7 7 0 0 0 8 8 8 0 0 0 0 0
```

样例 6：

```
Answer:  0,0;8,1;2,1;6,1;9,3;11,3;0,6;0,7;5,8;5,6
The number of unassigned vessels is: 0
The total waiting time is: 7
The last departure time is: 10
The total cost is: 24

input 'y' to show the breth or input other to skip y
1 1 1 1 1 0 7 8 8 8
1 1 1 1 1 0 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 3 3 3 3 3 7 8 8 8
0 3 3 3 3 3 10 10 9 9
0 4 4 4 4 4 10 10 9 9
0 4 4 4 4 4 10 10 9 9
0 2 2 0 0 0 10 10 9 9
0 2 2 5 5 5 5 5 9 9
0 2 2 5 5 5 5 5 0 0
0 2 2 6 6 6 6 6 0 0
```

样例 8：

```
Answer:  0,0;4,12;0,7;5,2;4,10;8,12;7,9;6,7;6,9;1,7;5,7;0,10
The number of unassigned vessels is: 0
The total waiting time is: 38
The last departure time is: 16
The total cost is: 92

input 'y' to show the breth or input other to skip y
1 1 1 1 1 1 1 3 3 3 12 12 12 12 12 12
1 1 1 1 1 1 1 10 10 10 12 12 12 12 12 12
1 1 1 1 1 1 1 10 10 10 12 12 12 12 12 12
1 1 1 1 1 1 1 10 10 10 12 12 12 12 12 12
1 1 1 1 1 1 1 10 10 10 5 5 2 2 2 2
0 0 4 4 4 4 4 11 11 11 5 5 2 2 2 2
0 0 4 4 4 4 4 8 8 9 5 5 2 2 2 2
0 0 4 4 4 4 4 8 8 7 7 7 2 2 2 2
0 0 4 4 4 4 4 8 8 7 7 7 6 6 6 6
0 0 4 4 4 4 4 8 8 7 7 7 6 6 6 6
```

样例 5：

```
Answer:  0,0;1,0;5,0;6,0;1,1;2,1;0,2;3,2;10,3;6,3;11,3;0,8;0,4;7,4;3,5;10,7;5,8
The number of unassigned vessels is: 0
The total waiting time is: 8
The last departure time is: 9
The total cost is: 25

input 'y' to show the breth or input other to skip y
1 1 7 7 13 13 13 13 12
2 5 7 7 13 13 13 13 12
2 6 7 7 13 13 13 13 12
2 6 8 8 8 15 15 15 12
2 0 8 8 8 15 15 15 12
3 3 8 8 8 15 15 15 17
4 4 4 10 0 15 15 15 0
4 4 4 10 14 14 14 14 14
4 4 4 10 14 14 14 14 14
4 4 4 10 14 14 14 14 14
0 0 0 9 9 9 9 16 16
0 0 0 11 11 11 11 16 16
```

样例 7：

```
Answer:  0,0;2,0;4,0;4,4;-1,-1;8,1;5,5;0,5;8,2;10,2;0,3;8,5;8,7;8,8
The number of unassigned vessels is: 1
The total waiting time is: 12
The last departure time is: 10
The total cost is: 134

input 'y' to show the breth or input other to skip y
1 0 0 11 11 8 8 8 8 0
1 0 0 11 11 8 8 8 8 0
2 2 2 2 2 8 8 8 8 0
2 2 2 2 2 8 8 8 8 0
3 3 3 3 4 8 8 8 8 0
3 3 3 3 4 7 7 7 7 7
3 3 3 3 4 7 7 7 7 7
3 3 3 3 4 7 7 7 7 7
0 6 9 9 4 12 12 13 14 14
0 6 9 9 0 12 12 13 14 14
0 6 10 10 10 10 0 13 14 14
0 6 10 10 10 10 0 13 0 0
```

样例 12：

```
Running Time : 213798ms

Answer:  1,0;0,0;2,0;5,2;6,2;0,6;1,5;5,8;10,5;0,11;1,11;4,12;7,12;8,12;8,14;10,1
3;9,16;-1,-1;9,15;6,24;0,23;11,17;4,19;11,23
The number of unassigned vessels is: 1
The total waiting time is: 49
The last departure time is: 29
The total cost is: 227

input 'y' to show the breth or input other to skip y
2 2 2 2 2 2 6 6 6 6 6 10 10 10 10 10 10 10 10 0 0 0 0 21 21 21 21 21 21 0
1 1 1 0 0 7 6 6 6 6 6 11 11 11 11 11 11 11 11 11 11 11 11 21 21 21 21 21 21 0
3 3 3 3 3 7 6 6 6 6 6 11 11 11 11 11 11 11 11 11 11 11 11 21 21 21 21 21 21 0
3 3 3 3 3 7 6 6 6 6 6 11 11 11 11 11 11 11 11 11 11 11 11 21 21 21 21 21 21 0
3 3 3 3 3 7 6 6 6 6 6 0 12 12 12 12 12 12 12 23 23 23 23 23 23 23 23 23 0 0
0 0 4 4 4 4 0 0 8 8 8 8 12 12 12 12 12 12 12 23 23 23 23 23 23 23 23 23 0 0
0 0 5 5 5 5 5 5 8 8 8 8 12 12 12 12 12 12 12 0 0 0 0 0 20 20 20 0 0 0
0 0 5 5 5 5 5 5 8 8 8 8 13 13 13 13 13 13 13 13 13 0 0 0 20 20 20 0 0 0
0 0 5 5 5 5 5 5 8 8 8 8 14 14 15 15 15 15 15 15 15 15 15 15 20 20 20 0 0 0
0 0 5 5 5 5 5 5 8 8 8 8 14 14 0 19 17 17 17 17 17 17 0 0 20 20 20 0 0 0
0 0 0 0 0 9 9 9 9 9 9 9 9 16 16 19 17 17 17 17 17 17 0 0 20 20 20 0 0 0
0 0 0 0 0 9 9 9 9 9 9 9 9 16 16 19 0 22 22 22 22 22 0 24 24 24 24 0 0 0
```

5.实验总结与心得

这一次的 Project 非常的有趣，就跟老师说的一样，想要做出来很简单，但是想要做好的话却很难。因为我比较笨，也想不到有什么又好，又快的算法去解决。首先的第一反应其实就是使用搜索去寻找最优解了，但是显然使用搜索的话，对于一些复杂的样例来说花费的时间太长了。我曾经尝试过使用深度优先搜索去做本次的作业，然而可能真的是我太菜了，在跑第 5 个样例的时候就用了十几二十分钟的时间。然而我又很菜，因此就先使用了最简单的贪心算法去完成。在中间遇到了不少的 bug，排除了以后做出来的结果跟题目中给出来的贪心的结果是一样的。但是我又不可能直接这样子就完成了这一次的作业，所以我决定在贪心的基础上做一些我能做得到的修改。但是从哪里入手呢？我决定从老师给出的样例中找出一些特殊情况来进行改进。首先我在第一个样例中发现了一种优化的方案：在港口书比较少的时候，注意要把一些比较短的船放在同一个港口，不同的时间，从某种角度这样就可以节省一定的空间。

但是这种优化我却无从下手，想了很久这个思路做不出来，因此就放弃了，从第二个样例开始入手。我在第二个样例中也发现了一个可以优化的方案，而且这一个更加容易去实现：调整一些服务时间比较长的船，尽量往边沿的泊位靠，是的能在中间部分留下较长的连续的空位去摆放别的船。所以按照这一个思路，我花了挺长的时间去修改贪心算法，终于达到了理想的效果。

然而，做出来一个算是有一点点进步的算法以后，又出现了一个问题了：这种改进仅仅是针对某些特殊的情况，但是也有可能会让其他的一些样例的解变得更加糟糕。因此为了减少这种情况的发生，我决定分别使用贪心算法和改进的贪心算法分别去求一次解，然后取最优的解输出。所以我的程序最后的成型就是这样了。

在 DDL 到来之前，我反省了一下自己，好像只做贪心真的是太水了，而且自己连第一个样例都没有摆满，实在是太过分了。为了做弥补，我又重新多写了一个搜索算法，终于把一些比较简单的样例摆满了，而且总花费也比较让我满意，可惜的是有一些还不能摆满，而且耗时还很长。

算法设计这一门课实在是太难了。显然，学习算法就是为了能够得到一个高效率的方法去求出最优的解，但是这个的难度显然是很大的。以后还是要继续好好的努力学习算法啊！要走的路还有很长。不仅仅要做多做题，还要多思考，积累多一点的知识，才能在这方面做得更好！

附录、提交文件清单

源程序文件：Project.cpp
Project_2.cpp
实验报告：算法设计 Project 实验报告.pdf
测试文件 1：test case.txt
测试文件 2：Project.xlsx