《面向对象程序设计与训练》实验报告

信息 学院 智能科学与技术 专业 2019 级

实验时间 2020 年 12 月 8 日

姓名 郑广敏 学号 20191130038

实验名称 CSP认证2017年9月测试真题

实验成绩

一、实验目的

1. 理解Java语言是如何体现面向对象编程基本思想，
2. 了解类的封装方法，以及如何创建类和对象，
3. 了解成员变量和成员方法的特性。
4. 了解类的继承性和多态性的作用。

二、实验仪器设备及软件

Windows2000 或XP，JDK1.6与Jcreator4.0

三、实验方案

1.注意两端只有一个邻居，其余部分均为b[i]=(a[i-1]+a[i]+a[i+1])/3

2.假如一个人的某个时段的结束时间比另一个人的结束时间要短，那么说明这个人的下一个时段也可能存在与另一个人同一段时间的交叠部分，因此继续将这个人的下一个时间段与另一个人的同一段时间相比，如此重复，直到其中一人的时间段都比较完了。

3.对于单个的标签或者id查询是很容易实现的，难点在于如何处理后代选择器。这里有两个思路，第一个构造树形结构，这样很容易就能查到与当前点相关的子孙；我采用的是第二种思路，首构造一个文档每行内容的一个数据结构，要包含三个属性（级别、标签和id），根据题目提示的贪心策略，对于当前行，其后级别不大于他的都是他的子孙。按照这个思路我设置了一个答案ans数组和缓冲buf数组，遍历每一级查询，buf中存的是当前级别上一级查询的结果，逐个取出buf中的行号，从这个行号可是遍历查询他的子孙是否有当前要查询的东西。

注意重复以及查询和文档的标签要全部转换为小写（大写）。

4.策略为DFS搜索，知道前两个数后可以确定第三个数的很少的有限可能，例如，除了两端之外，对于，如果已经确定了和，那么只有三种可能，即3\*--、3\*--+1、3\*--+2。对于起始端，有2\*-、2\*-+1两种可能，而对于起始点的取值范围则是[1，2\*]，每个点仅有少量可能，因此只要从起始点开始搜索每个点的状态，到最后一个点进行验证即可得到答案，但是当n较大时，搜索状态会很多，会超时，这里有80分。

一种改进的想法是进行剪枝，因为已知前两个数，如果后续不能成立，说明在该天这两个数的组合永远不可能满足要求，当再次在该天遇到这两个数的组合时，可以不向下搜索，但是不知道为什么，我在JAVA上写上剪枝优化后，时间效率反而降低了，这里只有60分。

5.太难，没啥思路，只能想出20分的暴力方法

四、实验步骤

1.

1. **import** java.util.\*;
2. **public** **class** Main {
3. **private** **static** Scanner sc;
4. **public** **static** **void** main(String[] args) {
5. sc=**new** Scanner(System.in);
6. String readData=sc.nextLine();
7. **int** n=Integer.parseInt(readData);
8. String[] linetwo=sc.nextLine().split(" ");
9. **int**[] price=**new** **int**[n];
10. **for**(**int** i=0;i<n;i++) {
11. price[i]=Integer.parseInt(linetwo[i]);
12. }
13. **int**[] newPrice=**new** **int**[n];
14. **for**(**int** i=0;i<n;i++) {
15. **int** cnt=1,num1=0,num2=0,num3=price[i];
16. **if**(i-1>=0) {
17. cnt++;
18. num1=price[i-1];
19. }
20. **if**(i+1<n) {
21. cnt++;
22. num2=price[i+1];
23. }
24. newPrice[i]=(num1+num2+num3)/cnt;
25. }
26. **for**(**int** i=0;i<n;i++) {
27. System.out.print(newPrice[i]);
28. **if**(i!=n-1) {
29. System.out.print(" ");
30. }
31. }
32. }
33. }

2.

1. **import** java.util.Scanner;
2. **public** **class** Main {
3. **private** **static** Scanner sc;
4. **public** **static** **void** main(String[] args) {
5. sc=**new** Scanner(System.in);
6. String readData=sc.nextLine();
7. **int** n=Integer.parseInt(readData);
8. **int**[] a=**new** **int**[n];
9. **int**[] b=**new** **int**[n];
10. **int**[] c=**new** **int**[n];
11. **int**[] d=**new** **int**[n];
12. **for**(**int** i=0;i<n;i++) {
13. String[] line=sc.nextLine().split(" ");
14. a[i]=Integer.parseInt(line[0]);
15. b[i]=Integer.parseInt(line[1]);
16. }
17. **for**(**int** i=0;i<n;i++) {
18. String[] line=sc.nextLine().split(" ");
19. c[i]=Integer.parseInt(line[0]);
20. d[i]=Integer.parseInt(line[1]);
21. }
22. **int** ans=0;
23. **int** i=0,j=0;
24. **while**(j<n && i<n){
25. **int** l=a[i]>c[j]?a[i]:c[j];
26. **int** r=0;
27. **if**(b[i]<d[j]) {
28. r=b[i];
29. i++;
30. }**else** {
31. r=d[j];
32. j++;
33. }
34. **if**(l<r) {
35. ans+=r-l;
36. }
37. }
38. System.out.print(ans);
40. }
41. }

3.

1. **import** java.util.\*;
2. **class** shuzu{
3. **public** **static** **boolean** is\_exist(**int** a[],**int** n,**int** len) {
4. **for**(**int** i=0;i<len;i++) {
5. **if**(a[i]==n) **return** **true**;
6. }
7. **return** **false**;
8. }
9. }
10. **class** Wen{
11. **public** **int** level;
12. **public** String id,name;
13. **public** Wen() {
14. **super**();
15. // TODO Auto-generated constructor stub
16. }
17. **public** Wen(**int** level, String id, String name) {
18. **super**();
19. **this**.level = level;
20. **this**.id = id;
21. **this**.name = name.toLowerCase();
22. }
23. }
24. **public** **class** Main {
25. **private** **static** Scanner sc;
26. **public** **static** **void** main(String[] args) {
27. sc=**new** Scanner(System.in);
28. **int** n=sc.nextInt();
29. **int** m=sc.nextInt();
30. **int** max\_level=0;
31. Wen a[]=**new** Wen[n];
32. String line=sc.nextLine();
33. **for**(**int** i=0;i<n;i++) {
34. line=sc.nextLine();
35. **int** cnt=0;
36. **while**(line.charAt(cnt)=='.') {
37. cnt++;
38. }
39. **int** level=cnt/2;
40. max\_level=level<max\_level?max\_level:level;
41. StringBuffer name=**new** StringBuffer(),id=**new** StringBuffer();
42. **for**(;cnt<line.length();cnt++) {
43. **if**(line.charAt(cnt)==' ') **break**;
44. name.append(line.charAt(cnt));
45. }
46. **for**(;cnt<line.length();cnt++) {
47. id.append(line.charAt(cnt));
48. }
49. a[i]=**new** Wen(level,id.toString(),name.toString());
50. }
51. **for**(**int** i=0;i<m;i++) {
52. String[] tmp=sc.nextLine().split(" ");
53. **int** ans[]=**new** **int**[n+1];
54. ans[0]=-1;
55. **int** p=0;//遍历指针
56. **int** num=0;
57. **int** is\_first=1;
58. **int** cur=0;//当前级别
59. **for**(**int** j=0;j<tmp.length;j++) {
60. **if**(tmp[j].charAt(0)!='#') tmp[j]=tmp[j].toLowerCase();
61. **int** cnt=0;
62. **if**(is\_first==0) {
63. **int** buf[]=ans.clone();
64. ans[0]=-1;
65. **for**(**int** k=0;buf[k]!=-1;k++) {
66. p=buf[k]+1;
67. cur=a[buf[k]].level;
68. **while**(p<n) {
69. **if**(cur>=a[p].level) **break**;
70. **if**(tmp[j].charAt(0)=='#') {//id
71. **int** pos=a[p].id.indexOf(tmp[j]);
72. **if**(pos!=-1) {
73. **if**(!shuzu.is\_exist(ans,p,cnt))
74. ans[cnt++]=p;
75. }
76. }**else** {//name
77. **int** pos=a[p].name.indexOf(tmp[j]);
78. **if**(pos!=-1) {
79. **if**(!shuzu.is\_exist(ans,p,cnt))
80. ans[cnt++]=p;
81. }
82. }
83. p++;
84. }
85. }
87. }**else** {
88. ans[0]=-1;
89. **while**(p<n) {
90. **if**(tmp[j].charAt(0)=='#') {//id
91. **int** pos=a[p].id.indexOf(tmp[j]);
92. **if**(pos!=-1) ans[cnt++]=p;
93. }**else** {//name
94. **int** pos=a[p].name.indexOf(tmp[j]);
95. **if**(pos!=-1) ans[cnt++]=p;
96. }
97. p++;
98. }
99. is\_first=0;
100. }
101. ans[cnt]=-1;
103. }
105. **for**(**int** q=0;ans[q]!=-1;q++) {
106. num++;
107. }
108. System.out.printf("%d ",num);
109. **for**(**int** q=0;q<num;q++) {
110. System.out.printf("%d ",ans[q]+1);
111. }
112. System.out.println();
113. }
114. }
115. }

4.（80分）

1. **import** java.util.\*;
2. **public** **class** q4 {
3. **private** **static** Scanner sc;
4. **public** **static** **boolean** dfs(**int** a[],**int** b[],**int** n,**int** i) {
5. **if**(i==n-1) {
6. **if**(b[i] == (a[i]+a[i-1])/2 ) {
7. **for**(**int** j=0;j<n;j++) {
8. System.out.print(a[j]+" ");
9. }
10. System.out.println();
11. **return** **true**;
12. }**else** **return** **false**;
13. }**else** {//其他一般情况的推导
14. **int** tmp=b[i]\*3-a[i]-a[i-1];
15. a[i+1]=tmp;
16. **if**(a[i+1]>0)
17. **if**(dfs(a,b,n,i+1))
18. **return** **true**;
19. a[i+1]=tmp+1;
20. **if**(a[i+1]>0)
21. **if**(dfs(a,b,n,i+1))
22. **return** **true**;
23. a[i+1]=tmp+2;
24. **if**(a[i+1]>0)
25. **if**(dfs(a,b,n,i+1))
26. **return** **true**;
27. }
28. **return** **false**;
29. }
30. **public** **static** **void** main(String[] args) {
31. sc=**new** Scanner(System.in);
32. **int** n=sc.nextInt();
33. sc.nextLine();
34. **int**[] a=**new** **int**[n];
35. **int**[] b=**new** **int**[n];
36. **for**(**int** i=0;i<n;i++) {
37. b[i]=sc.nextInt();
38. }
39. **for**(**int** i=1;i<=2\*b[0];i++) {
40. a[0]=i;
41. **int** tmp=b[0]\*2-a[0];
42. a[1]=tmp;
43. **if**(q4.dfs(a,b,n,1)) **break**;
44. a[1]=tmp+1;
45. **if**(q4.dfs(a,b,n,1)) **break**;
46. }
47. }
48. }

5.（20分）

1. **import** java.util.\*;
2. **public** **class** q5 {
3. **private** **static** Scanner sc;
4. **public** **static** **void** main(String[] args) {
5. sc=**new** Scanner(System.in);
6. **int** m=sc.nextInt(),l=sc.nextInt(),r=sc.nextInt();
7. sc.nextLine();
8. **long**[] a=**new** **long**[r+1];
9. **long**[] k=**new** **long**[100010];
10. **for**(**int** i=1;i<=m;i++) {
11. k[i]=sc.nextLong();;
12. }
13. a[0]=1;
14. **for**(**int** i=1;i<=r;i++) {
15. **int** min=(i<m ? i : m);
16. **for**(**int** j=1;j<=min;j++) {
17. a[i]=(a[i]+k[j]\*a[i-j]%998244353)%998244353;
18. }
19. }
20. **for**(**int** i=l;i<=r;i++) {
21. System.out.println(a[i]);
22. }
23. }
24. }

D:\桌面\截图_20200008060058.png

五、实验结果及分析

1.



2.



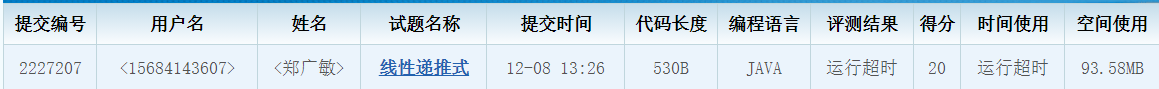
3.



4.

D:\桌面\截图_20200008060058.png

5.



六、实验总结及体会

1.在第二题中，不能想当然地将两人时间段一对一地比较，要考虑极端情况

2.第三题，标签对大小写不敏感，不仅是指文档，也是指查询都要转换大小写，此外利用这种贪心的策略（树形结构思路也一样）会出现查询div p时会将div div p认为是两个div p，这样就会出现重复，要注意去重。

3.第四题，看似状态众多，但是除了缩小边界外，还可以根据数学关系以及已知的数来缩小状态数，这样就可以用dfs进行搜素了

七、教师评语