**2022-10-28 算法作业**

1. n个数，找出其中最小的k个数，写出代码，要求最坏情况下的时间复杂度不能高于O(nlog2k)。

**测试用例：**

6 3

89 78 1 -9 0 67

**输出：**

-9 0 1

**代码：**

1. #include<iostream>
2. #include<algorithm>
3. using namespace std;
4. int main(){
5. int n, k;
6. while (cin >> n >> k){
7. int \*a = new int [n];
8. for(int i = 0; i < n; i++){
9. cin >> a[i];
10. }
11. sort(a,a+n);
12. for(int i = 0; i < k; i++){
13. cout << a[i] <<" ";
14. }
15. cout << endl;
16. delete(a);
17. }
19. return 0;
20. }

**运行结果：**

图片包含 图表

描述已自动生成

**时间复杂度：**

1. 写程序输出8皇后问题的所有排列，要求使用非递归的深度优先遍历。

**测试用例：**

无

**输出：**

8皇后所有排序

**代码：**

1. #include <iostream>
2. #include <math.h>
3. using namespace std;
4. #define n 8
5. int q[n] = {-1};      *//第i行上的皇后在第几列*
6. int cnt = 0;       *//计数器，多少种解*
7. void print();       *//输出方法*
8. bool check(int j);      *//位置是否合法*
9. void queen();       *//皇后放置*
11. int main(){
12. queen();
13. cout << "共有"<< cnt << "种布局。" << endl;
14. return 0;
15. }
16. void print(){       *//输出方法*
17. int i,j;
18. for(i = 0; i < n; i++){
19. for(j = 0; j < n; j++){
20. if(q[i] == j)
21. cout << " Q ";
22. else
23. cout << " □ ";
24. }
25. cout<<endl;
26. }
27. cout<<endl;
28. }
29. bool check(int j){      *//位置是否合法*
30. int i=0;
31. while(i<j){
32. if(q[i] == q[j] || abs(q[i]-q[j]) == abs(i-j))
33. return false;
34. i++;
35. }
36. return true;
37. }
38. void queen(){       *//皇后放置*
39. int i = 0;
40. while(i >= 0){      *//直到回溯到第一行也找不到可以放置皇后的位置，函数结束*
41. q[i]++;
42. while(!check(i) && q[i]<n)
43. q[i]++;      *//找到这一行能够放置皇后的列数*
44. if(q[i] < n){
45. if(i == n-1){    *//最后一行也成功放置皇后，找到一个解*
46. cnt++;
47. print();
48. }
49. else
50. i++;
51. }
52. else{
53. q[i] = -1;     *//第i行到最后也没有找到合适的位置，初始化本行的列数*
54. i--;      *//回到上一行，从下一列开始寻找*
55. }
56. }
57. }

**运行结果：**

表格

中度可信度描述已自动生成**（仅展示部分）**

**时间复杂度：**

1. 有n个作业，a1,a2，…，an，作业aj的处理时间为tj，产生的效益为pj，最后完成期限为dj，作业一旦被调度则不能中断，如果作业aj在dj前完成，则获得效益pj，否则无效益。给出最大化效益的作业调度算法。

**测试用例：**

T = 0, 1, 4, 1

D = 0, 5, 4, 5

P = 0, 2, 8, 6

**输出：**

**14**

**代码：**

1. #include <iostream>
2. #include <algorithm>
3. using namespace std;
4. #define MAXN 51
5. int n = 3;
6. *// num  time    deadline    produce*
7. int t[] = {0,1,4,1}; *//  1     1        5           2*
8. int d[] = {0,5,4,5}; *//  2     4        4           8*
9. int p[] = {0,2,8,6}; *//  3     1        5           6*
10. int bestp = 0;
11. struct Action{
12. int time;
13. int deadline;
14. int produce;
15. bool operator < (const Action t) const{
16. return produce > t.produce;
17. }
18. };
19. Action A[MAXN];
20. void solve();
21. int main(){
22. for (int i = 1; i <= n; i++){
23. A[i].time = t[i];
24. A[i].deadline = d[i];
25. A[i].produce = p[i];
26. }
27. solve();
28. cout<<"最优解："<< bestp << endl;
29. }
30. */// @brief 主要判定就是当前的任务的deadline 是否 >= sum + 当前任务时间；*
31. */// @brief 例子，当前deadline 为 5，sum = 4; 而完成的当前任务时间为8，那么 4 + 8 > 5 ， 不合符条件*
32. void solve(){
33. sort(A+1, A+n+1);
34. int sum = 0;
35. for (int i = 1; i <= n; i++){
36. if (A[i].deadline >= sum + A[i].time){                 *//若sum + A[i].time超过了限制，则不能计入*
37. bestp += A[i].produce;
38. sum += A[i].time;
39. }
40. }
41. }

**运行结果：**

图形用户界面, 文本

描述已自动生成

**时间复杂度：**