**计算机与科学技术学院2024-2025(2) 学期**

**算法专业竞赛**

**（时间：90分钟）**

**一、单选题（每题3分，共30分）**

1. 以下哪个算法的时间复杂度为 O(nlogn)？（ B ）

A. 冒泡排序

B. 快速排序（平均情况）

C. 插入排序

D. 选择排序

2. 在分治法中，递归式 T(n)=kT(n/m)+f(n) 中 f(n) 表示：（ B ）

A. 子问题规模

B. 分解和合并的代价

C. 递归深度

D. 子问题数量

3. 以下哪个不是回溯法的典型应用？（ C ）

A. N皇后问题

B. 子集和问题

C. 单源最短路径问题（Dijkstra）

D. 图着色问题

4. 回溯法可以通过什么策略避免无效搜索？（ A ）

A. 限界函数剪枝

B. 递归关系函数剪枝

C. 随机剪枝

D. BackTrack函数剪枝

5. 若 T (n)=2T (n/2)+nlogn，则 T (n) 的阶是（ C ）

A. O (n)

B. O (nlogn)

C. O (n (logn)2)

D. O (n2)

6.对 n 个元素的数组进行排序，若采用归并排序，则最坏情况下的时间复杂度为（ B ）

A. O (n)

B. O (nlogn)

C. O (n2)

D. O (2n)

7. 以下关于 NP 完全问题的说法，错误的是（ C ）

A. NP 完全问题是 NP 类问题中最难的问题

B. 假设一个 NP 完全问题能在多项式时间内得到解决，那么所有 NP 类中的所有问题都可以在多项式时间内得到解决

C. 旅行商问题是一个P类问题

D. 目前尚未找到求解 NP 完全问题的多项式时间算法

8. 以下关于最小生成树的说法，错误的是（ D ）

A. Prim算法和Kruskal算法都是求解最小生成树的贪心算法

B. Prim算法的时间复杂度为 O (n2)，适用于稠密图

C. Kruskal算法的时间复杂度为 O (eloge)，适用于稀疏图

D. 一个连通图的最小生成树是唯一的

9. 以下关于舍伍德算法的说法，正确的是（ A ）

A. 舍伍德算法是一种非确定性随机化算法

B. 舍伍德算法的平均性能与最坏情况性能相同

C. 舍伍德算法的最坏情况下的性能优于平均性能

D. 舍伍德算法适用于所有问题

10. 快速排序的partition划分操作时间复杂度是：（ D ）

A. O(1)

B. O(logn)

C. O(nlogn)

D. O(n)

**二、填空题（每空2分，共20分）**

1. 哈夫曼编码问题：假设有5个字符的频率分布如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| 17 | 8 | 7 | 12 | 56 |

请给出字符A的编码：（ 00/11/10/01 ），字符A的编码种数一共有（ 4 ）种。//任一均给分

请给出哈夫曼编码的平均码长：（ 1.86 ）。

2. 以下是一段递归函数的代码：

#include <iostream>

int recursive(int n) {

if (n <= 1) {

return 1;

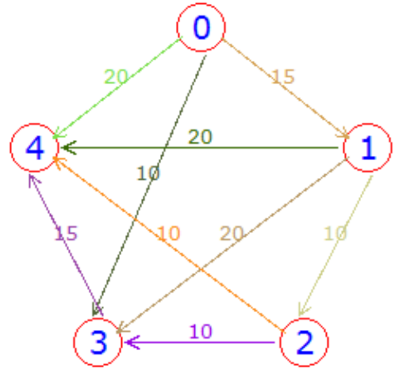
} else {

return recursive(n - 2) + recursive(n - 4);

}

}

上述代码空间复杂度为（ O(n) ），时间复杂度为（ 1.27n及以上）。假设n=12，则该函数返回值为（ 20 ）。



四、程序分析题（每空3分，共30分）

1. 变形的最大相容活动集合问题：某国有个大型会议中心，其中有n个活动要安排，要求是在相容活动数量最大的前提下，这些相容活动的总时间要最长。以下是求解该问题的C++代码实现，请补充完整。

|  |  |
| --- | --- |
| #include <iostream>  **(1) #include <vector>**  #include <algorithm>  using namespace std;  struct Activity {  int start; //开始时间  int end; //结束时间  int duration; //活动时长  };  // 比较函数：按结束时间排序  bool compare(const Activity& a, const Activity& b) {  return a.end < b.end;  }  int main() {  vector<Activity> activities = {  {1, 4, 3}, {3, 5, 2}, {0, 6, 6}, {5, 7, 2},  {3, 9, 6}, {5, 9, 4}, {6, 10, 4}, {8, 11, 3},  {8, 12, 4}, {2, 14, 12}, {12, 16, 4}  }; // 输入  // 按结束时间排序  sort(activities.begin(), activities.end(), compare);    **（2）int n =activities.size();**  vector<int> mtt(n, 0); // mtt[i] 表示前i个活动的最大总时间  vector<int> prev(n, -1); // 前一个相容活动的索引    // 预处理prev数组：找到每个活动之前最远的相容活动  for (int i = 0; i < n; ++i) {  for (int j = i-1; j >= 0; --j) {  if (activities[j].end <= activities[i].start) {  **（3）prev[i] = j;**  break;  }  }  }  //计算最大总时间  mtt[0] = activities[0].duration;  for (int i = 1; i < n; ++i) {  int include = activities[i].duration;  if (prev[i] != -1) include += mtt[prev[i]];  int exclude = mtt[i-1];  **(4)mtt[i] = max(include, exclude);**  } | // 输出结果  cout << "最大总时间: " << mtt[n-1] << endl;    // 回溯找出选中的活动  vector<int> selected;  int i = n-1;  while (i >= 0) {  if (i == 0 || mtt[i] != mtt[i-1]) {  selected.push\_back(i);//该活动被选中  i = prev[i];  } else {  **(5)i--;**  }  }    // 输出选中的活动  cout << "选中的活动（按结束时间排序）:" << endl;  for (auto it = selected.rbegin(); it != selected.rend(); ++it) {  cout << "[" << activities[\*it].start << ", "  << activities[\*it].end << "] 持续时间: "  << activities[\*it].duration << endl;  }  return 0;  } |

上述代码采用了哪种算法来求解？（6）动态规划

按照程序代码给出的活动集合输入，则变形的最大相容活动集合选中的是哪几个活动？

（7） [1, 4, 3], [5, 9, 4] ,[8, 12, 4], [12, 16, 4]

这些活动的总时间是多少？（8） 15

2、以下代码是采用分治法测试一个字符串是否是回文字符串（如：level，refer等都是回文字符串，即正反读的字符序列结果都一样），请补充完整。

|  |  |
| --- | --- |
| #include <iostream>  #include <string>  using namespace std;  // 分治法判断子串s是否是回文，left和right分别是左边和右边索引位置  bool isPalindrome(const string& s, int left, int right) {  if (**（1）left >= right** ) {  return true;  }    if (s[left] != s[right]) {  return false;  }  // 递归检查  **（2） return isPalindrome(s, left + 1, right - 1);**  }  // 主函数：判断整个字符串是否是回文  bool isPalindrome(const string& s) {  return isPalindrome(s, 0, s.length() - 1);  } | int main() {  string input;  cout << "请输入一个字符串: ";  cin >> input;  if (isPalindrome(input)) {  cout << "\"" << input << "\" 是回文字符串。" << endl;  } else {  cout << "\"" << input << "\" 不是回文字符串。" << endl;  }  return 0;  } |

**五、程序设计题（20分）**

1. 网络信息时代，除了手机和视频之外，还真是无聊啊，学习是不可能学习的，必须到了谈恋爱的时候了。浙江公园大学的小明和小亮都比较腼腆，虽然都有自己心仪的女神，但迟迟不敢表白。这天，在寝室长小建的建议下，小明和小亮决定玩一个游戏，谁输了谁就在下周一11:45分的家和堂门口大声喊：“XXX,我喜欢你”520秒。游戏规则如下：由出主意的寝室长小建把寝室玩了1年的扑克牌随机抽一些分到n个盒子里，然后小明和小亮轮流（第1次由小明执行）进行如下操作：

* 选择一个装有*ai*张牌的盒子 *i*（即 *ai*>*0*），从中拿走 1 张牌（*ai* 减 1）。
* 如果没有有效的盒子可选择，当前玩家输掉游戏。
* 如果在操作后，最大值与最小值的差超过 *k*，即 *max* (*a₁,a₂,…,an*)−*min* (*a₁,a₂,…,an*)>*k*，则当前玩家（做出最后一步移动的玩家）输掉游戏。

由于两人比较腼腆，双方都决定要采取最优策略不能输掉游戏，请你编程预测游戏结束后，谁会站在下周一11:45分的家和堂门口表白（不考虑数据量和内存大小）。

输入（input.txt）：

* 第一行：测试用例数 *t*。
* 每个测试用例：
  + 第一行：两个整数 *n* 和 *k*。
  + 第二行：*n* 个整数 *a₁,a₂,…,an*。

输出（output.txt）：对每个测试用例，输出 “xiaoming is sick for love”（不含引号）表示小明去表白，否则输出 “xiaoliang is sick for love”。

|  |  |
| --- | --- |
| 输入示例：  3  3 1  2 1 2  3 1  1 1 3  2 1  1 4 | 输出示例：  xiaoliang is sick for love  xiaoliang is sick for love  xiaoming is sick for love |

评分标注：贪心算法，include语句3分，main函数1分，文件输入输出2分，while主循环2分，第一个for循环5分，第2个for循环5分，输出结果2分。

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

ifstream fin("input.txt");

ofstream fout("output.txt");

int t;

fin >> t;

while (t--) {

int n, k;

fin >> n >> k;

vector<int> a(n);

int min\_val = 1e9 + 7;

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

fin >> a[i];

min\_val = min(min\_val, a[i]);

}

long long total = 0;

for (int x : a) {

total += x - min\_val;

}

if (total % 2 == 1) {

fout << "xiaoliang is sick for love!" << endl;

} else {

fout << "xiaoming is sick for love!" << endl;

}

}

fin.close();

fout.close();

return 0;

}