初赛资料与真题模拟 · 程序填空

2018 Junior

完善程序

（最大公约数之和）下列程序想要求解整数n的所有约数两两之间最大公约数的和对 10007 求余后的值，试补全程序。（第一空2 分，其余 3 分）举例来说，4 的所有约数是 1, 2, 4 。1 和 2 的最大公约数为 1 ； 2 和 4 的最大公约数为 2 ； 1 和 4 的最大公约数为 1 。于是答案为1 + 2 + 1 = 4。

要求getDivisor 函数的复杂度为$O(\sqrt{n})$，gcd 函数的复杂度为$O(\log \max(a, b))$。

```
#include <iostream>
using namespace std;

const int N = 110000, P = 10007;
int n;
int a[N], len;
int ans;

void getDivisor() {
    len = 0;
    for (int i = 1; ___________ <= n; ++i)
        if (n % i == 0) {
          a[++len] = i;
          if ( _________ != i) a[++len] = n / i;
        }
}

int gcd(int a, int b) {
    if (b == 0) { ___________________ ;  }
    return gcd(b, _______________);
}

int main() {
    cin >> n;
    getDivisor();
    ans = 0;
    for (int i = 1; i <= len; ++i)
        for (int j = i + 1; j <= len; ++j)
            ans = ( __________ ) % P;
    cout << ans << endl;
    return 0;
}

```

- 答案：

1. i * i;
2. n / i
3. return a
4. a % b
5. ans + gcd(a[i],a[j])

- 分析：
  - 对于第四、五空较为简单，直接默写欧几里得辗转相除法即可。
  - getDivisor函数，顾名思义，为求因子的函数。其中`if (n % i == 0) a[++len] = i;`说明for循环枚举数字i是否是n的因子。
  - 由于因子总是成对出现，所以只需要枚举到$\sqrt{(n)}$即可，题目也是如此要求。
  - 因此，若n是完全平方数，则要避免因子i与n/i的重复添加。
  - a[]数组保存了各个因子，枚举，求和，即可得到答案。

对于一个 1 到 n 的排列 P（即 1 到 n 中每一个数在 P 中出现了恰好一次），令 q[i] 为第 i 个位置之后第一个比 P[i] 值更大的位置，如果不存在这样的位置，则 q[i] = n + 1。举例来说，如果 n = 5 且 P 为 1 5 4 2 3，则 q 为2 6 6 5 6。

下列程序读入了排列 P，使用双向链表求解了答案。试补全程序。

```
#include <iostream>
using namespace std;

const int N = 100010;
int n;
int L[N], R[N], a[N];

int main() {
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        int x; cin >> x;
        _____________ ;
    }
    for (int i = 1; i <= n; ++i) {
        R[i] = _____________ ;
        L[i] = i - 1;
    }
    for (int i = 1; i <= n; ++i) {
        L[ _____________ ] = L[a[i]];
        R[L[a[i]]] = R[ _____________ ];
    }
    for (int i = 1; i <= n; ++i)
        cout << _____________ << " ";
    cout << endl;
    return 0;
}
```

- 答案：

1. a[x]=i;
2. i+1;
3. R[a[i]]
4. a[i]
5. R[i]

- 分析：
  - 由于是排列，故我们需要记录数字为i的位置是a[i]。这样方便我们处理大小比较关系。
  - 题目说明，采用双向链表，因此必然有前驱后继的指针。又由于使用数组模拟，故每个结点的前驱就是i-1,后继就是i+1.分别保存在L数组与R数组。
  - 模仿上下文，对称书写，构造双向链表。

2018 Senior

提高组完善程序的第一题即为普及组的第二题。

一只小猪要买 N 件物品（N 不超过 1000）。

它要买的所有物品在两家商店里都有卖。第 i 件物品在第一家商店的价格是 a[i]，在第二家商店的价格是 b[i]，两个价格都不小于 0 且不超过 10000。如果在第一家商店买的物品的总额不少于 50000，那么在第一家店买的物品都可以打 95 折(价格变为原来的 0.95 倍)。

求小猪买齐所有物品所需最少的总额。

输入：第一行一个数 N。接下来N 行，每行两个数。第 i 行的两个数分别代表 a[i]，b[i]。
输出：输出一行一个数，表示最少需要的总额，保留两位小数。

试补全程序。

```
#include <cstdio>
#include <cstdlib>
using namespace std;

const int Inf = 1000000000;
const int threshold = 50000;
const int maxn = 1000;

int n, a[maxn], b[maxn];
bool put_a[maxn];
int total_a, total_b;

double ans;
int f[threshold];

int main() {
    scanf("%d", &n);
    total_a = total_b = 0;
    for (int i = 0; i < n; ++i) {
        scanf("%d%d", a + i, b + i);
        if (a[i] <= b[i]) total_a += a[i];
        else total_b += b[i];
    }
    ans = total_a + total_b;
    total_a = total_b = 0;
    for (int i = 0; i < n; ++i) {
        if (_____________) {
            put_a[i] = true;
            total_a += a[i];
        } else {
            put_a[i] = false;
            total_b += b[i];
        }
    }
    if ( _____________ ) {
        printf("%.2f", total_a * 0.95 + total_b);
        return 0;
    }
    f[0] = 0;
```

```
    for (int i = 1; i < threshold; ++i)
        f[i] = Inf;
    int total_b_prefix = 0;
    for (int i = 0; i < n; ++i)
        if (!put_a[i]) {
            total_b_prefix += b[i];
            for (int j = threshold - 1; j >= 0; --j) {
                if ( _____________ >= threshold && f[j] != Inf)
                    ans = min(ans, (total_a + j + a[i]) * 0.95
+_____________);
                f[j] = min(f[j] + b[i], j >= a[i] ? ___________________: Inf);
            }
        }
    printf("%.2f", ans);
    return 0;
}
```

- 答案：

1. a[i] * 0.95 <= b[i]
2. total_a >= threshold / total_a >= 50000
3. total_a + j + a[i]
4. f[j] + total_b - total_b_prefix
5. f[j - a[i]]

- 分析：
  - 贪心思路，一开始比较每个货物，选择便宜的一家购买，价格累加到total_a或total_b。
  - 接着，考虑更换商家是否会更赚——如果这个物品在A商店购买打95折比在B商店便宜，则考虑在A商店选购。否则直接在B商店采购。
  - 如果满足A商店打折条件——满50000则说明之前的决策是最优的。
  - 若无法满足，则说明当前情况下无法买够50000，则需要考虑其他方案。
  - 设f[i] 表示从 当前前缀状态下， 买总价格为 i 的物品，所需要去B 商场的最小花费，此时转变成一个类似背包的动态规划问题。枚举每个本来在B商店购买的商品放到A商店购买，能否凑够50000获得折扣，使得总花费降低。
  - 也即，对于每个物品，考虑是直接在B商品购买还是在A商店购买（保证凑够50000）时更优。

2017 Junior

（快速幂） 请完善下面的程序，该程序使用分治法求$x^p modm$ 的值。（第一空 2 分，其余 3 分）

输入：三个不超过 10000 的正整数 x，p，m。
输出：$x^p modm$的值。
提示：若 p 为偶数，$x^p = (x^2)^{p/2}$；若 p 为奇数，$x^p = x * (x^2)^{(p-1)/2}$

```
#include<iostream>
using namespace std;
int x, p, m, i, result;
int main(){
    cin >> x >> p >> m;
    result = _________________;
    while (_____________________){
        if (p % 2 == 1)
```

```
            result = ____________________;
        p /= 2;
        x = ____________________;
    }
    cout << ____________________ << endl;
    return 0;
}
```

- 答案：

1. 1
2. p>0 或 p!=0 或 p
3. result * x % m
4. x * x % m
5. result

- 分析：裸快速幂，非递归模板。背诵。

（切割绳子） 有 n 条绳子，每条绳子的长度已知且均为正整数。绳子可以以任意正整数长度切割，但不可以连接。现在要从这些绳子中切割出 m 条长度相同的绳段，求绳段的最大长度是多少。（第一、二空 2.5 分，其余 3 分）

输入：第一行是一个不超过 100 的正整数 n，第二行是 n 个不超过10^6的正整数，表示每条绳子的长度，第三行是一个不超过10^8的正整数 m。 输出：绳段的最大长度，若无法切割，输出 Failed。

```
#include<iostream>
using namespace std;
int n, m, i, lbound, ubound, mid, count;
int len[100]; // 绳子长度
int main(){
    cin >> n;
    count = 0;
    for (i = 0; i < n; i++){
        cin >> len[i];
        ____________________________;
    }
    cin >> m;
    if (____________________________){
        cout << "Failed" << endl;
        return 0;
    }
    lbound = 1;
    ubound = 1000000;
    while (____________________________){
        mid = ____________________;
        count = 0;
        for (i = 0; i < n; i++)
            ____________________;
        if (count < m) ubound = mid - 1;
        else lbound = mid;
    }
    cout << lbound << endl;
    return 0;
}
```

- 答案：

1. count += len[i]
2. count < m
3. lbound < rbound
4. (lbound + rbound + 1) /2
5. count += len[i]/mid

- 分析：
  - 若所有段加起来都没有m，则必然无解。（最小情况分成m个1）。
  - 使用count累加求和，判断与m的大小。
  - 二分答案的条件判断，多带几组值验证即可。
  - 对于每个答案进行验证：看每段能分解成多少个mid的绳段，使用count累加，判断能否满足条件，

2017 Senior

（大整数除法）给定两个正整数p和q，其中p不超过10^{100}， q不超过100000， 求 p 除以 q 的商和余数。（第一空 2 分，其余 3 分） 输入：第一行是 p 的位数 n，第二行是正整数 p，第三行是正整数 q。输出：两行，分别是 p 除以 q 的商和余数。

```
#include <iostream>
using namespace std;
int p[100];
int n, i, q, rest;
char c;
int main(){
    cin >> n;
    for (i = 0; i < n; i++){
        cin >> c;
        p[i] = c - '0';
    }
    cin >> q;
    rest = ____________________;
    i = 1;
    while (_________________ && i < n){
        rest = rest * 10 + p[i];
        i++;
    }
    if (rest < q) cout << 0 << endl;
    else{
        cout << ____________________;
        while (i < n){
            rest = ____________________;
            i++;
            cout << rest / q;
        }
        cout << endl;
    }
    cout << ____________________ << endl;
    return 0;
}
```

- 答案：

1. p[0]

2. rest < q
3. rest / q
4. rest % q * 10 + p[i]
5. rest % q

- 分析：模拟竖式除法，rest表示余数。每次用rest找到一个大于q的部分，采用秦九韶算法。

（最长路径）给定一个有向无环图，每条边长度为 1，求图中的最长路径长度。（第五空 2 分，其余 3 分） 输入：第一行是结点数 n（不超过 100）和边数 m，接下来 m 行，每行两个整数 a， b，表示从结点 a 到结点 b 有一条有向边。结点标号从 0 到(n-1)。 输出：最长路径长度。 提示：先进行拓扑排序，然后按照拓扑序计算最长路径。

```
#include <iostream>
using namespace std;
int n, m, i, j, a, b, head, tail, ans;
int graph[100][100]; // 用邻接矩阵存储图
int degree[100];     // 记录每个结点的入度
int len[100];        // 记录以各结点为终点的最长路径长度
int queue[100];      // 存放拓扑排序结果
int main(){
    cin >> n >> m;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            graph[i][j] = 0;
    for (i = 0; i < n; i++)
        degree[i] = 0;
    for (i = 0; i < m; i++){
        cin >> a >> b;
        graph[a][b] = 1;
        ____________________________;
    }
    tail = 0;
    for (i = 0; i < n; i++)
        if (____________________________){
            queue[tail] = i;
            tail++;
        }
    head = 0;
    while (tail < n - 1){
        for (i = 0; i < n; i++)
            if (graph[queue[head]][i] == 1){
                ____________________________;
                if (degree[i] == 0){
                    queue[tail] = i;
                    tail++;
                }
            }
        ____________________________;
    }
    ans = 0;
    for (i = 0; i < n; i++){
        a = queue[i];
        len[a] = 1;
        for (j = 0; j < n; j++)
            if (graph[j][a] == 1 && len[j] + 1 > len[a])
                len[a] = len[j] + 1;
```

```
        if (_____________________)
            ans = len[a];
    }
    cout << ans << endl;
    return 0;
}
```

- 答案：

1. degree[b]++
2. degree[i]==0
3. degree[i]--
4. head++
5. ans<len[a]

- 分析
    - 拓扑排序中很重要的一部分是顶点度数的处理，该部分只能放在读入时初始化。
    - 用bfs的方式处理拓扑排序，数组模拟队列，每次去掉结点后将下一步到达的点度数-1
    - 没有出度的顶点是路径的终点，记录对应的len值，取最大值即为答案。

2016 Junior

读入整数。请完善下面的程序，使得程序能够读入两个 int 范围内的整数， 并将这两个整数分别输出，每行一个。（第一、五空 2.5 分，其余 3 分）
输入的整数之间和前后只会出现空格或者回车。输入数据保证合法。
例如：
输入：
123 -789
输出： 132
-789

```
#include <iostream>
using namespace std;

int readint(){
    int num = 0;           // 存储读取到的整数
    int negative = 0;     // 负数标识
    char c;                // 存储当前读取到的字符
    c = cin.get();
    while ((c < '0' || c > '9') && c != '-')
        c = ____________________;
    if (c == '-') negative = 1;
    else ____________________;
    c = cin.get();
    while (____________________){
        ____________________;
        c = cin.get();
    }
    if (negative == 1) ____________________;
    return num;
}
int main(){
    int a, b;
    a = readint();
```

```
    b = readint();
    cout << a << endl << b << endl;
    return 0;
}
```

- 答案:

1. cin.get()
2. num = c-'0'
3. c>='0' && c<= '9'
4. num = num * 10 + c-'0'
5. num = -num;

- 分析： 字符串快速读入，或称输入输出外挂。背诵。

（郊游活动）有 n 名同学参加学校组织的郊游活动，已知学校给这 n 名同学 的郊游总经费为 A 元，与此同时第 i 位同学自己携带了 Mi 元。为了方便郊 游，活动地点提供 B(≥n)辆自行车供人租用，租用第 j 辆自行车的价格为 Cj 元，每位同学可以使用自己携带的钱或者学校的郊游经费，为了方便账务管 理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多 有多少位同学能够租用到自行车。（第四、五空 2.5 分，其余 3 分）
本题采用二分法。对于区间[l, r]，我们取中间点 mid 并判断租用到自行 车的人数能否达到 mid。判断的过程是利用贪心算法实现的。

```
#include <iostream>
using namespace std;
#define MAXN 1000000

int n, B, A, M[MAXN], C[MAXN], l, r, ans, mid;

bool check(int nn) {
    int count = 0, i, j;
    i = ____________________;
    j = 1;
    while (i <= n) {
        if(_________________)
            count += C[j] - M[i];
        i++;
        j++;
    }
    return ____________________;
}

void sort(int a[], int l, int r) {
    int i = l, j = r, x = a[(l + r) / 2], y;
    while (i <= j) {
        while (a[i] < x) i++;
        while (a[j] > x) j--;
        if (i <= j) {
            y = a[i]; a[i] = a[j]; a[j] = y;
            i++; j--;
        }
    }
if (i < r) sort(a, i, r);
if (l < j) sort(a, l, j);
}
```

```
33
34  int main() {
35      int i;
36      cin >> n >> B >> A;
37      for (i = 1; i <= n; i++)
38          cin >> M[i];
39      for (i = 1; i <= B; i++)
40          cin >> C[i];
41      sort(M, 1, n);
42      sort(C, 1, B);
43      l = 0;
44      r = n;
45      while (l <= r) {
46          mid = (l + r) / 2;
47          if(__________________){
48              ans = mid;
49              l = mid + 1;
50          }else
51              r = ____________________;
52      }
53      cout << ans << endl;
54      return 0;
55  }
```

- 答案：

1. n-nn+1
2. M[i]<=C[j]
3. count<=A
4. check(mid)
5. mid-1

- 分析：
  - check 函数是用来判断租车人数能否到达mid，也即nn. 模拟即可。
  - 二分的题目需要注意边界的变化方式与判定条件。

2016 Senior

（交朋友）根据社会学研究表明，人们都喜欢找和自己身高相近的人做朋友。现在有 n 名身高两两不相同的同学依次走入教室，调查人员想预测每个人在 走入教室的瞬间最想和已经进入教室的哪个人做朋友。当有两名同学和这名 同学的身高差一样时，这名同学会更想和高的那个人做朋友。比如一名身高为 1.80 米的同学进入教室时，有一名身高为 1.79 米的同学和一名身高为1.81米的同学在教室里，那么这名身高为 1.80 米的同学会更想和身高为 1.81 米的同学做朋友。对于第一个走入教室的同学我们不做预测。由于我们知道所有人的身高和走进教室的次序，所以我们可以采用离线的做法来解决这样的问题，我们用排序加链表的方式帮助每一个人找到在他之前进入教室的并且和他身高最相近的人。（第一空 2 分，其余 3 分）

```
1  #include <iostream>
2  using namespace std;
3  #define MAXN 200000
4  #define infinity 2147483647
5  int answer[MAXN], height[MAXN], previous[MAXN], next[MAXN];
6  int rank[MAXN];
7  int n;
```

```
void sort(int l, int r){
    int x = height[rank[(l + r) / 2]], i = l, j = r, temp;
    while (i <= j){
        while (height[rank[i]] < x) i++;
        while (height[rank[j]] > x) j--;
        if ((_____________________)){
            temp = rank[i];
            rank[i] = rank[j];
            rank[j] = temp;
            i++;
            j--;
        }
    }
    if (i < r) sort(i, r);
    if (l < j) sort(l, j);
}
int main(){
    cin >> n;
    int i, higher, shorter;
    for (i = 1; i <= n; i++){
        cin >> height[i];
        rank[i] = i;
    }
    sort(1, n);
    for (i = 1; i <= n; i++){
        previous[rank[i]] = rank[i - 1];
        (_____________________);
    }
    for (i = n; i >= 2; i--){
        higher = shorter = infinity;
        if (previous[i] != 0)
            shorter = height[i] - height[previous[i]];
        if (next[i] != 0) (_____________________);
        if ((_____________________)) answer[i] = previous[i];
        else answer[i] = next[i];
        next[previous[i]] = next[i];
        (_____________________);
    }
    for (i = 2; i <= n; i++)
        cout << i << ":" << answer[i];
    return 0;
}
```

- 答案：

1. i<=j
2. next [rank[i]]=rank[i+1]
3. higher = height[next[i]] - height[i]
4. shorter < higher
5. previous[next[i]] = previous[i]

- 分析：
    - 采用类似快排的方式，以第i个人作为基准，将人群分为两类。
    - 用数组模拟链表，建立链表的next数组。
    - 分别计算高一点与矮一点的身高差距，选择更接近的那个。

(交通中断）有一个小国家，国家内有 n 座城市和 m 条双向的道路，每条道路连接着两座不同的城市。其中 1 号城市为国家的首都。由于地震频繁可能导致某一个城市与外界交通全部中断。这个国家的首脑想知道，如果只有第i(i>1)个城市因地震而导致交通中断时，首都到多少个城市的最短路径长度会发生改变。如果因为无法通过第 i 个城市而导致从首都出发无法到达某个城市，也认为到达该城市的最短路径长度改变。 对于每一个城市 i，假定只有第 i 个城市与外界交通中断，输出有多少个 城市会因此导致到首都的最短路径长度改变。 我们采用邻接表的方式存储图的信息，其中 head[x]表示顶点 x 的第一条 边的编号，next[i]表示第 i 条边的下一条边的编号，point[i]表示第 i 条边的终点，weight[i]表示第 i 条边的长度。（第一空 2 分，其余 3 分）

```
#include <iostream>
#include <cstring>
using namespace std;
#define MAXN 6000
#define MAXM 100000
#define infinity 2147483647
int head[MAXN], next[MAXM], point[MAXM], weight[MAXM];
int queue[MAXN], dist[MAXN], visit[MAXN];
int n, m, x, y, z, total = 0, answer;
void link(int x, int y, int z){
    total++;
    next[total] = head[x];
    head[x] = total;
    point[total] = y;
    weight[total] = z;
    total++;
    next[total] = head[y];
    head[y] = total;
    point[total] = x;
    weight[total] = z;
}
int main(){
    int i, j, s, t;
    cin >> n >> m;
    for (i = 1; i <= m; i++){
        cin >> x >> y >> z;
        link(x, y, z);
    }
    for (i = 1; i <= n; i++)
        dist[i] = infinity;
    ____________________;
    queue[1] = 1;
    visit[1] = 1;
    s = 1;
    t = 1;
    // 使用 SPFA 求出第一个点到其余各点的最短路长度
    while (s <= t){
        x = queue[s % MAXN];
        j = head[x];
        while (j != 0){
            if (____________________){
                dist[point[j]] = dist[x] + weight[j];
                if (visit[point[j]] == 0){
                    t++;
                    queue[t % MAXN] = point[j];
                    visit[point[j]] = 1;
                }
```

```
            }
            j = next[j];
        }
        ____________________________;
        s++;
    }
    for (i = 2; i <= n; i++){
        queue[1] = 1;
        memset(visit, 0, sizeof(visit));
        visit[1] = 1;
        s = 1;
        t = 1;
        while (s <= t){ // 判断最短路长度是否不变
            x = queue[s];
            j = head[x];
            while (j != 0){
                if (point[j] != i && ____________________ && visit[point[j]]
== 0){
                    ____________________________;
                    t++;
                    queue[t] = point[j];
                }
                j = next[j];
            }
            s++;
        }
        answer = 0;
        for (j = 1; j <= n; j++)
            answer += 1 - visit[j];
        cout << i << ":" << answer - 1 << endl;
    }
    return 0;
}
```

- 答案：

1. dist[1] = 0
2. dist[x]+weight[j]<dist[point[j]]
3. visit[x]=0
4. weight[j]+dist[x]==dist[point[j]]
5. visit[point[j]]=1

- 分析：SPFA算法模板，背诵。