パソコン甲子園 2010

【予選問題＊解説】

問題 01 水道料金を節約しよう

問題のポイント

問題文で述べられた仕様を理解し、その通りに動作するプログラムを記述できるかを問う問題です。変数、入出力、四則演算に加え、条件分岐や繰り返し処理についての知識が必要です。

問題の解き方

いくつかのアルゴリズムが考えられますが、 w が 100 以下と小さい値なので、水量を 1m^3 ずつ増やして料金を加算していく簡単なアルゴリズムでも大丈夫です。具体的には、使用水量 w を引数として水道料金を計算する関数 getCost() を作ります。getCost() では cost を基本料金である 1150 に初期化し、累計水量が 11 から w m^3 について 1m^3 あたりの料金を cost に加算していきます。加算する料金はそのときの累計水量 i によって 120, 140, あるいは 160 と変化させます。

main 関数ではデータセットの処理を行うループの中で w を読み込み、4280 - getCost(w) を出力します。

講評

提出数：688、正解数 304

提出数も多く、不正解の理由も多岐に渡りましたが、入力のミス、出力形式のエラー（必要の無い改行や空白を出力している等)、単一のデータセットしか処理できていない等、問題の内容に関連するものではなく、問題を解く際の基本的な事項に関するミスが多く見られました。これらの理由で不正解になってしまうのはもったいないので、自分の環境で実行し確認してから提出しましょう。

また、問題特有の間違いとしては、10 の剰余算を取ることで各段階の水の量を計算しているプログラムが、10、20、30 の 10 で割り切れてしまうケースで撃墜されているプログラムも多く見られました。

解答例(C++)

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

int getCost(int w){
   int cost = 1150;                        // 第１段階
   for (int i = 11; i <= w; i++ ){
      if ( i <= 20 ) cost += 125;          // 第２段階
      else if ( 30 < i ) cost += 160;      // 第４段階
      else cost += 140;                    // 第３段階
   }
   return cost;
```

```
}

int main(){
   for( int w; cin >> w && w >= 0; ) {
      cout << 4280 - getCost(w) << endl;
   }
   return 0;
}
```

解答例(Java)

```
import java.util.Scanner;
public class P01{

   public P01(){
       Scanner scan = new Scanner(System.in)
       for(int w; scan.hasNextInt() && w >= 0;){
         System.out.println( 4280-getCost(w));
       }
   }

   public int getCost(int w){
       int cost = 1150;                  // 第１段階
       for (int i = 11; i <= w; i++ ){
          if ( i <= 20 ) cost += 125;     // 第２段階
          else if ( 30 < i ) cost += 160; // 第４段階
          else cost += 140;               // 第３段階
       }
       return cost;
   }

   public static void main(String[] args){
      P01 p01 = new P01();
   }
}
```

問題 02 病院でウォーキング

問題のポイント

与えられた情報の中から、ある項目が最大である要素を探索するプログラムが記述できるかを問う問題です。

問題の解き方

現在の最大の距離を maxv，その最大の距離を歩いた患者の番号を maxp とします。最初に maxv を -1 に初期化し、それぞれの患者の情報 p，d1，d2 を読み込み、d1 + d2 が maxv よりも大きければ maxv を d1 + d2 に、maxp を p に置き換えます。データセットについて全ての患者の情報を調べた後の maxp 及び maxv を出力します。

講評

提出数 : 542、正解数 : 179

不正解の解答では、単一のデータセットしか処理していない、出力形式ミス、最大値の初期化忘れ等が多く見られました。特に、変数の初期化は基本的なことなので、慣れるまでは毎回確認するように意識しましょう。

解答例(C++)

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

int main(){
   int n, p, maxp, maxv, d1, d2;

   while( cin >> n && n ){
      maxv = -1;
      for ( int i = 0; i < n; i++ ){
         cin >> p >> d1 >> d2;
         if ( maxv < d1 + d2 ){
            maxp = p;
            maxv = d1 + d2;
         }
      }
      cout << maxp << " " << maxv << endl;
   }

   return 0;
}
```

解答例(Java)

```
import java.util.Scanner;
```

```
public class P02 {
   private int n, p, maxp, maxv, d1, d2;

   public P02(){
      Scanner scan = new Scanner(System.in);
      while(true){
           n = scan.nextInt();
           if(n == 0) break;
           maxv = -1;
           for(int i = 0 ; i < n ; i++){
               p = scan.nextInt();
               d1 = scan.nextInt();
               d2 = scan.nextInt();
               if(maxv < d1+d2){
                  maxp = p;
                  maxv = d1 + d2;
               }
           }
           System.out.println(maxp+" "+maxv);
       }

   }
   public static void main(String[] args){
      new P02();
   }
}
```

問題 03 塾のクラス分け

問題のポイント

等価演算、比較演算、論理演算等を用いて簡単な条件分岐を行うプログラムが記述できるかを問う問題です。

問題の解き方

３科目それぞれの点数 m, e, j を引数として、クラスを表す文字を返す関数 getClass()を作成します。getClass は問題文で与えられた条件を if 文、等価演算、比較演算、論理演算を組み合わせ、該当するクラスの文字を返します。 main 関数ではそれぞれの点数 pm, pe, pj を読み込み getClass(pm, pe, pj)を出力します。

講評

提出数 : 459、正解数 : 240

不正解の解答には、問題 01、02 の講評に記載したミスに加え、条件式の判定で用いる境界条件のミスも多くありました。解答作成時は、入出力例のデータで確認するとともに、判定を左右しそうな境界条件を入力してみるなど、意識的に確認するようにしましょう。

解答例(C++)

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

char getClass(int m, int e, int j){
   if ( m == 100 || e == 100 || j == 100 ||
        (m + e)/2 >= 90 || (m + e + j)/3 >= 80 ) return 'A';
   if ((m + e + j)/3 >= 70 ||
        (m + e + j)/3 >= 50 && m >= 80 || e >= 80 ) return 'B';
   return 'C';
}

int main(){
   int n, pm, pe, pj;
   while( cin >> n && n ){
      for ( int i = 0; i < n; i++ ){
         cin >> pm >> pe >> pj;
         cout << getClass(pm, pe, pj) << endl;
      }
   }
   return 0;
}
```

解答例(Java)

```
import java.util.Scanner;
```

```
public class P03 {
   private static int n, pm, pe, pj;

   public P03() {
      Scanner scan = new Scanner(System.in);

      while (true) {
         n = scan.nextInt();
         if (n == 0)
            break;
         for (int i = 0; i < n; i++) {
            pm = scan.nextInt();
            pe = scan.nextInt();
            pj = scan.nextInt();
            System.out.println(getClass(pm, pe, pj));
         }
      }

   }

   char getClass(int m, int e, int j) {
      if (m == 100 || e == 100 || j == 100 || (m + e) / 2 >= 90
         || (m + e + j) / 3 >= 80)
         return 'A';
      if ((m + e + j) / 3 >= 70 || (m + e + j) / 3 >= 50 && m >= 80
         || e >= 80)
         return 'B';
      return 'C';
   }

   public static void main(String[] args) {
      new P03();
   }
}
```

問題 04 人気のアイスクリーム店

問題のポイント

配列に格納し、それらの値を適宜更新してゆき、最終的な値を棒グラフ状に整形して出力できるかを問う問題です。

問題の解き方

一般的には、大量のデータを効率良く記録、更新するにはデータベースなどのデータ構造を用いると便利です。でも、この問題のように値の数値が小さく総数も少ない場合は配列で十分です。まず、整数型配列 T[10] を用意します。T[x] に数字 x の出現回数を記録します。そして、各要素について、出現回数分だけ*を出力し、改行します。

講評

提出数：294、正解数：202

正解解答のほとんどが上記のアルゴリズムを採用していました。不正解の解答では、問題01～03と同様な基本的なミスが目立ちましたので、練習を通して慣れるようにしましょう。

解答例(C++)

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

int main(){
   int n, T[10], x;
   while( cin >> n && n ){
      for ( int i = 0; i < 10; i++ ) T[i] = 0;
      for ( int i = 0; i < n; i++ ){ cin >> x; T[x]++; }
      for ( int i = 0; i < 10; i++ ){
         if (T[i] == 0 ) cout << "-" << endl;
         else {
            for ( int j = 0; j < T[i]; j++ ) cout << "*";
            cout << endl;
         }
      }
   }
   return 0;
}
```

解答例(Java)

```
import java.util.Scanner;

public class P04 {
   private static int n, x;
   private static int[] T;
   public P04(){
```

```
        Scanner scan = new Scanner(System.in);
        T = new int[10];
        while (true) {
        n = scan.nextInt();
            if (n == 0)
                break;
            for ( int i = 0; i < 10; i++ ) T[i] = 0;
            for ( int i = 0; i < n; i++ ){ x=scan.nextInt(); T[x]++; }
            for ( int i = 0; i < 10; i++ ){
                if (T[i] == 0 ) System.out.println("-");
                else {
                    for ( int j = 0; j < T[i]; j++ )  System.out.print("*");
                    System.out.println();
                }
            }
        }
    }
    public static void main(String[] args) {
        new P04();
    }
}
```

問題 05 博士の愛した２進数

問題のポイント

基数変換の知識を用い、実数(浮動小数点数)に応用できるかを問う問題です。

問題の解き方

与えられた実数を整数にシフトして基数変換を行います。まず与えられた数に 16 を掛け 2 進数表現で 4 桁分シフトします。シフトした値を 16.0 で割ったものが x と異なる、またはシフトした値が 4096 以上(２進数表現で 13 桁以上)であれば、制限桁数に収まらないと判断できるので NA と出力します。桁数に収まる場合は基数変換を行い適切な位置に.を含め変換結果を出力します。

講評

提出数 : 213、正解数 : 47

不正解の解答では、出力形式の間違い、NA となる場合の判定ミス、基数変換操作のミスなどが主な原因でした。

解答例(C++)

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

void convert(int x, int p){
    if ( p == 12 ) return;
    convert(x/2, p+1);
    if ( p == 3 ) cout << ".";
    cout << x%2;
}

int main(){
    for ( double x; cin >> x && x >= 0; ){
        int shift = (int)(x*16);
        if ( x == shift/16.0 && shift < 4096 ){
            convert(shift, 0);
            cout << endl;
        } else {
            cout << "NA" << endl;
        }
    }
    return 0;
}
```

解答例(Java)

```
import java.util.Scanner;
public class P05 {
    private double n;
    public P05(){
```

```
        Scanner scan = new Scanner(System.in);
        while (true) {
            n = scan.nextDouble();
            if (n <= 0)
                break;
            int shift = (int)(n*16);
            if ( n == shift/16.0 && shift < 4096 ){
                convert(shift, 0);
                System.out.println();
            } else {
                System.out.println("NA");
            }
        }
    }

    void convert(int x, int p){
        if ( p == 12 ) return;
        convert(x/2, p+1);
        if ( p == 3 ) System.out.print(".");
        System.out.print(x%2);
    }
    public static void main(String[] args) {
        new P05();
    }
}
```

問題 06 FizzBuzz

問題のポイント

簡単なシミュレーション（状況の再現）を行うプログラムが記述できるかを問う問題です。

問題の解き方

可変長の配列として扱える vector<int> を使って、残っているプレイヤーを管理しシミュレーションを行います。まず、発言の順番 x と発言された言葉 y を引数にとり、発言が正答かどうかを判定する関数 judge() を定義します。この関数では問題に定義されたように、3 で割り切れかつ 5 で割り切れる場合では発言が"FizzBuzz"であるか、3 で割り切れる場合では発言が"Fizz"であるか、5 で割り切れる場合では発言が"Buzz"であるか、それ以外では順番 x が y に等しいかどうかを判定します。シミュレーション本体では、現在の発言者をカーソル cur で示し、1 つずつ発言 num を読み込み、発言が正答であれば cur を 1 つ進め、そうでなければ cur の位置のプレイヤーを削除します。カーソルは残っているプレイヤーを循環するように cur = cur%P.size() とし、途中でプレイヤーが一人になった場合は、読み込み処理を除いてシミュレーション処理を終了します。

講評

提出数：181、正解数：16

不正解の解答では、出力の文字と文字の間の半角スペース数が正しくない（多すぎる場合や足りない場合）、次のプレイヤーの選択ミスなどの間違いが見られました。

解答例(C++)

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

bool judge(int x, string y){
   if ( x % 3 == 0 && x % 5 == 0 ) return y == "FizzBuzz";
   if ( x % 3 == 0 ) return y == "Fizz";
   if ( x % 5 == 0 ) return y == "Buzz";
   if ( !isdigit(y[0]) ) return false;
   return x == atoi(y.c_str());
}

int main(){
   int m, n;
   string num;
   while( cin >> m >> n && m ){
      vector<int> P;
      int cur = 0, w = -1;
      for ( int i = 1; i <= m; i++ ) P.push_back(i);

      for ( int i = 1; i <= n; i++ ){
```

```
            cin >> num;
            if ( w != -1 ) continue;
            if ( judge(i, num) ) cur++;
            else P.erase(P.begin() + cur);
            cur = cur%P.size();
            if ( P.size() == 1 ) w = P[0];
        }

        if ( w != -1 ) cout << w << endl;
        else {
            for ( int i = 0; i < P.size(); i++ )
            cout << ((i)?" ":"") << P[i];
            cout << endl;
        }
    }
    return 0;
}
```

問題 07 四つ子素数

問題のポイント

素数表を作成し、高速な素数判定を行うプログラムが作成できるかを問う問題です。

問題の解き方

まずエラトステネスの篩によって入力値の最大値である 10,000,000 以下の数についての素数表を作成します。この素数表を使い四つ子素数の位置を n から開始する線形探索によって探し出します。

講評

提出数：226、正解数：22

不正解の解答では、エラトステネスの篩の実装ミス、素数判定方法の間違い、通常の素数判定（計算時間 O(n)）による時間制限エラー、素数表を入力のたびに作り直すことによる時間制限エラーなどが見られました。素数に関するアルゴリズムは基本ですので、素数判定の方法の種類とそれらの性質も含めて良く勉強しておきましょう。

解答例(C++)

```
#include<iostream>
using namespace std;
#define N 10000000

bool P[N + 1];

void eratos(){
   for ( int i = 0; i <= N; i++ ) P[i] = true;
   P[0] = P[1] = false;
   for ( int i = 2; i*i <= N; i++ )
      if ( P[i] )
         for ( int j = 2; i*j <= N; j++ ) P[i*j] = false;
}

int getQuads(int n){
   for ( int i = n; i >= 13; i--)
      if ( P[i] && P[i-2] && P[i-6] && P[i-8] ) return i;
}

int main(){
   eratos();
   int n;
   while( cin >> n && n ) cout << getQuads(n) << endl;
   return 0;
}
```

問題 08 迷子の双子

問題のポイント

探索アルゴリズムの理解と、状態遷移がからむ場合の応用ができるかを問う問題です。

問題の解き方

アルゴリズムの一つを説明します。状態を表すためにクラス State を定義します。１つの状態は二人の位置である(ty, tx)及び(ky, kx)で表します。その状態に移った時間を保持しておくために t も宣言します。１度訪問した状態を map で管理するため、State クラスには比較演算子を定義する必要があります。ここでは、二人の位置関係で異なる状態の大小関係を定義しています。この状態空間の中で幅優先探索を行い最短の時間を求めます。

講評

提出数: 38、正答数:5

不正解となった解答では、縦型探索（深さ優先探索など）で解こうとして時間制限エラーになったチームがいくつかありました。縦型と横型の探索の使い分けを判断するのは難しいところですが、状態を変化させながらその最小回数を求める問題の場合、幅優先探索などの横型の探索が向いている傾向があります。

解答例(C++)

```
#include<iostream>
#include<queue>
#include<map>
using namespace std;
#define MAX 50
#define INFTY (1<<10)

class State{
   public:
   int ty, tx, ky, kx, t;
   State(int ty=0, int tx=0, int ky=0, int kx=0, int t=0):
         ty(ty), tx(tx), ky(ky), kx(kx), t(t){}

   bool operator < ( const State &s) const{
      if ( ty != s.ty ) return ty < s.ty;
      if ( tx != s.tx ) return tx < s.tx;
      if ( ky != s.ky ) return ky < s.ky;
      if ( kx != s.kx ) return kx < s.kx;
      return false;
   }
};

int H, W, G[MAX][MAX];

bool isBlock(int y, int x){
   if ( 0 > y || 0 > x || y >= H || x >= W ) return true;
   return G[y][x];
```

```
}

int bfs(int ty, int tx, int ky, int kx){
    queue<State> Q;
    map<State, bool> V;
    State start = State(ty, tx, ky, kx, 0);
    V[start] = true;
    Q.push(start);

    State u, v;
    const int dy[4] = {0, -1, 0, 1};
    const int dx[4] = {1, 0, -1, 0};
    int nty, ntx, nky, nkx;

    while(!Q.empty()){
        u = Q.front(); Q.pop();
        if ( u.tx == u.kx && u.ty == u.ky ) return u.t;
        for ( int r = 0; r < 4; r++ ){
            nty = u.ty + dy[r];
            ntx = u.tx + dx[r];
            nky = u.ky + dy[r]*(-1);
            nkx = u.kx + dx[r]*(-1);
            if ( isBlock(nty, ntx) ){ nty = u.ty; ntx = u.tx; }
            if ( isBlock(nky, nkx) ){ nky = u.ky; nkx = u.kx; }
            v = State(nty, ntx, nky, nkx);
            if ( !V[v] ){
                v.t = u.t + 1;
                V[v] = true;
                Q.push(v);
            }
        }
    }
    return INFTY;
}

int main(){
    int tx, ty, kx, ky;
    while( cin >> W >> H && W ) {
        cin >> tx >> ty >> kx >> ky;
        for ( int i = 0; i < H; i++ )
        for ( int j = 0; j < W; j++ ) cin >> G[i][j];
        int t = bfs(ty-1, tx-1, ky-1, kx-1);
        if ( t == INFTY ) cout << "NA" << endl;
        else cout << t << endl;
    }
    return 0;
}
```

問題 09 自転車でダイエット

問題のポイント

拡張グラフにおいて最短経路(距離)を求めるプログラムが作成できるかを問う問題です。ただし、計算過程で負のコストが発生するため、アルゴリズムの選択には注意が必要です。

問題の解き方

この解答例では、すべてのケーキ屋さんに立ち寄ったかの情報を 5 ビット分の整数として表し、その整数と現在地の組をノードとしてグラフを表します。この拡張グラフにおいてベルマンフォードのアルゴリズムを適用します。

講評

提出数:19、 正答数:3

ダイクストラで解いているチームが多く存在しましたが、グラフが負のエッジを含む場合、ダイクストラでは最適な解を得られない場合が存在します。最短経路を求めるアルゴリズムはさまざまな種類が存在しますが、ベルマンフォードとダイクストラなどは基本ですので、まずはこれらを習得しておきましょう。

解答例(C++)

この解説ではスペース節約のため以下のマクロを用いています。

```
#define REP(i, a, b) for ( int i = a; i < b; i++ )
#define rep(i, n) REP(i, 0, n)
```

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
#define REP(i, a, b) for ( int i = a; i < b; i++ )
#define rep(i, n) REP(i, 0, n)
#define INFTY (1<<21)

int m, n, k, d, C[7], G[110][110], N, D[110][64];

int getID(string str){
   if ( str == "H" ) return 0;
   if ( str == "D" ) return N - 1;
   string num = str.substr(1, str.size()-1);
   if ( str[0] == 'C' ) return atoi(num.c_str());
   return atoi(num.c_str()) + m;
}

bool isC(int x){ return 1 <= x && x <= m; }

int bellman(){
   rep(i, N) rep(j, (1<<m)) D[i][j] = INFTY;
   D[0][0] = 0;
   rep(o, N*(1<<m)){
```

```
        bool change = false;
        rep(u, N) rep(v, N){
            if ( u == v || G[u][v] == INFTY ) continue;
            rep(b, (1<<m)){
                if (isC(v) && (b&(1<<(v-1))) != 0 ) continue;
                if ( isC(v) ) {
                    if ( D[u][b] + G[u][v] - C[v] < D[v][b|(1<<(v-1))] ) {
                        D[v][b|(1<<(v-1))] = D[u][b] + G[u][v] - C[v];
                        change = true;
                    }
                } else {
                    if ( D[u][b] + G[u][v] < D[v][b] ){
                        D[v][b] = D[u][b] + G[u][v];
                        change = true;
                    }
                }
            }
        }
        if ( !change ) break;
    }

    int minv = INFTY;
    rep(i, (1<<m)) minv = min(minv, D[N-1][i]);
    return minv;
}

int main(){
    string source, target;
    int cost;
    while(cin >> m >> n >> k >> d && m ){
        N = 2 + n + m;
        REP(i, 1, m+1) cin >> C[i];
        rep(i, N) rep(j, N) G[i][j] = INFTY;
        rep(i, d){
            cin >> source >> target >> cost;
            int s = getID(source);
            int t = getID(target);
            G[t][s] = G[s][t] = min(G[s][t], cost * k);
        }
        cout << bellman() << endl;
    }
    return 0;
}
```

問題 10 こぶたぬきつねこ

問題のポイント

有向グラフのオイラー閉路の有無を判定するプログラムを作成できるかを問う問題です。

問題の解き方

まず、与えられた単語で有向グラフを作成します。グラフはアルファベットの数である 26 個のノードを持ち、与えられた各単語について最初の文字が始点、最後の文字が終点となるようにエッジを追加します。このグラフにおいて、あるノードから全てのエッジをちょうど一回通り元のノードに戻る閉路（オイラー閉路）があるかどうかを判定します。有向グラフがオイラー閉路を持つ条件は、このグラフの全てのノードについて、入力エッジと出力エッジの数が等しいことです。ただし、グラフが連結で無い場合はすべての単語で循環しりとりを行うことは不可能になるのでグラフの連結成分の数が 1 つになることも判定しなければなりません。

講評

提出数:140、正答数:4

多くの提出がありましたが正答率は非常に少ない問題でした。不正解の解答の多くは独自の手法で解こうとして失敗していました。限られた時間で正確なプログラムを作るには、確立された既存のアルゴリズムを用いるのが無難です。この問題のように、問題文の意図を読み取り、どのアルゴリズムを適用できるかを考えるようにしましょう。

解答例(C++)

この解説ではスペースの節約のために以下のマクロを利用しています。

```
#define rep(i, n) for ( int i = 0; i < n; i++)
```

```
#include<iostream>
#include<string>
using namespace std;
#define rep(i, n) for ( int i = 0; i < n; i++)

int n, in[26], out[26], G[26][26];
bool visited[26];

bool check(){
   rep(i, 26) if ( in[i] != out[i] ) return false;
   return true;
}

void dfs(int u){
   visited[u] = true;
   rep(v, 26) if ( !visited[v] && G[u][v] ) dfs(v);
}
```

```
int getNComponent(){
    int nc = 0;
    rep(i, 26){
        if ( in[i] == 0 && out[i] == 0 ) continue;
        if ( !visited[i] ){ nc++; dfs(i); }
    }
    return nc;
}

int main(){
    while( cin >> n && n ){
        rep(i, 26){
            in[i] = out[i] = 0;
            visited[i] = false;
            rep(j, 26) G[i][j] = 0;
        }

        string str;
        rep(i, n){
            cin >> str;
            in[str[0]-'a']++;
            out[str[str.size()-1]-'a']++;
            G[str[0]-'a'][str[str.size()-1]-'a'] = 1;
            G[str[str.size()-1]-'a'][str[0]-'a'] = 1;
        }

        cout << (( check() && getNComponent() == 1 )?"OK":"NG") << endl;
    }
    return 0;
}
```