Интерфейс:

Программа работает в текстовом режиме. В начале игры вводится количество столбиков и максимальное количество столбиков, которое можно взять первым ходом. Количество монет в каждом столбике выбирается случайным образом. Затем компьютер и игрок по очереди делают ходы, до тех пор, пока все столбики не будут взяты.

Алгоритм:

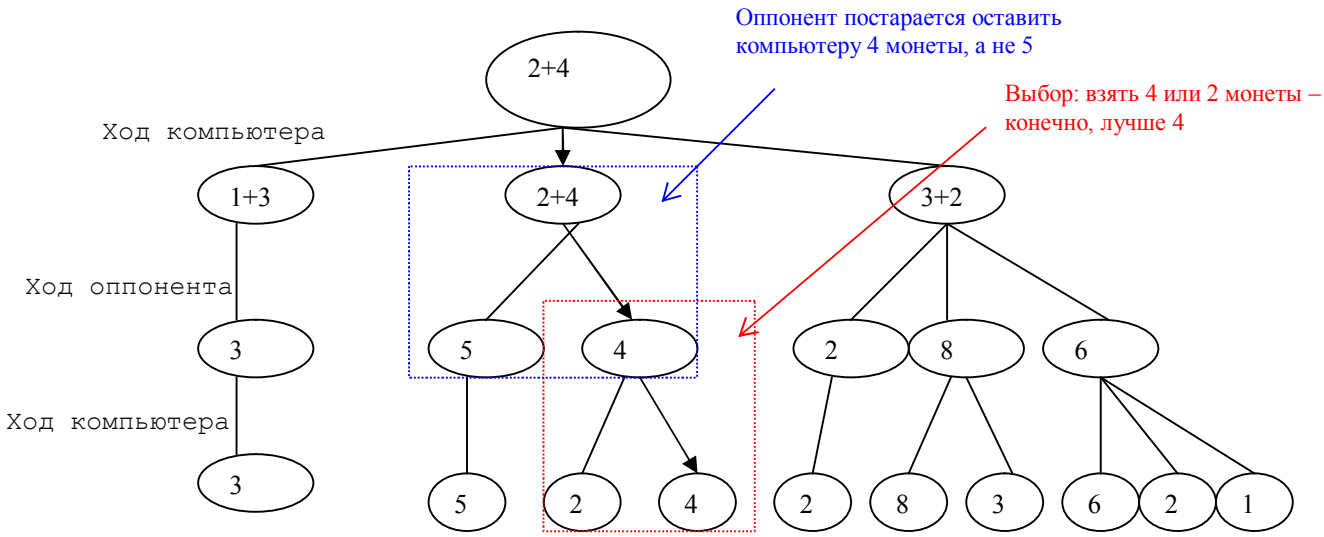
Для выбора хода компьютер перебирает все возможные варианты продолжения игры, просматривая «дерево вариантов».

1. Общий принцип - компьютер своими ходами старается взять монет побольше и считает, что то же самое будет делать и противник, оставляя компьютеру как можно меньше.
2. Когда компьютеру нужно выбрать из нескольких вариантов своего хода (т.е. взять один столбик, два или три) – он выбирает вариант с наибольшей оценкой. Оценка выбранного варианта становится оценкой хода.
3. Для вычисления оценки своего варианта компьютер:
   а) рассматривает возможные ответные ходы оппонента.
   б) Затем для каждого них считает оценку своего последующего хода (как считает – см. выше) (если последующий свой ход невозможен потому что столбики кончатся, то оценка=0).
   в) Из получившихся оценок выбирается наименьшая (почему - см. пункт 1) и к ней прибавляется кол-во монет, которое возьмёт компьютер исследуемым вариантом своего хода. Получившаяся сумма становится оценкой варианта.
4. Если при рассмотрении варианта своего хода выясняется, что после такого хода столбиков больше не останется (или останется один, который оппонент, естественно, следующим ходом заберёт), то кол-во монет, которое возьмёт компьютер исследуемым вариантом своего хода и становится оценкой варианта.

Таким образом, процесс идёт так:

В свою очередь хода компьютер выбирает вариант с наибольшей оценкой и делает ход сообразно выбранному варианту. Чтобы выбрать вариант, ему придётся рассмотреть все варианты, а для каждого из них – все варианты последующего хода оппонента, а для них в свою очередь – варианты своих ходов, и так далее. То есть просматриваются все варианты продолжения игры до самого конца.

Этот процесс проще объяснить на графическом его изображении. Рассмотрим пример – дерево всех вариантов продолжения игры.



В этом примере компьютер должен сделать выбор из 3-х вариантов хода (1-й ряд) – один столбик, два столбика, три столбика. Для этого он для каждого варианта рассматривает варианты ответных ходов оппонента (2-й ряд).

Например, если взять два столбика, то оппонент затем может взять один или два (выделено синим). Но считается, что он возьмёт два. Почему? Потому что если он возьмёт один, то следующим ходом компьютер возьмёт 5 монет, а если оппонент возьмёт два столбика – то следующими ходом компьютер сможет взять максимум 4 монеты – значит оппоненту выгоднее взять два столбика.

Таким образом, для первоначальной ситуации вариант «2 столбика» получает оценку 6: 4 – это наименьшая из оценок для ходов оппонента, а 2 – столько монет в этих самых двух столбиках, которые этим вариантом и берутся.

Для варианта «1 столбик» оценка получается равной 4, а для варианта «3 столбика» - равной 5. Вариант 2 имеет наибольшую оценку, следовательно компьютер выберет его

Программная реализация

Алгоритм реализован в виде программы на языке С. Помимо основной, программа имеет ещё 3 функции:

Функция create вызывается в начале работы для задания массива чисел, который и представляет собой столбики монет. Параметр – количество столбиков (чисел в массиве)

Функция take используется, чтобы подсчитать, сколько монет возьмёт компьютер тем или иным ходом. Параметры – количество имеющихся столбиков, количество столбиков, которое надо взять. Возвращаемое значение – суммарное количество монет во взятых столбиках.

Рекурсивная функция check осуществляет подсчёт и выбор оценок. Параметры – кол-во оставшихся столбиков, вариант хода компьютера.

Внутри функции check – два вложенных цикла – внешний (i) перебирает варианты ходов оппонента, а внутренний (j) – последующих ходов компьютера, вызывая функцию check для каждого из них и выбрав наибольшее возвращённое значение. Из всех значений, выбранных внутренним циклом, внешний выбирает наименьшее. К выбранному значению прибавляется кол-во монет, взятых ходом, для которого эта функция была вызвана. Полученное значение возвращается как результат работы функции. Если при вызове функции выясняется, что после хода, для которого функция была вызвана, дальнейшие ходы уже невозможны (либо возможен только ход оппонента со взятием 1 столбика), то возвращается просто количество монет, взятых этим ходом, без вызова циклов.

В главной функции программы процесс игры осуществляет цикл по условию. В нём сначала выбирает вариант хода компьютер (выбирая тот ход, для которого функция check вернёт наибольшее значение) – ход делается: удаляется соответствующее количество столбиков и монетки из них добавляются к сумме компьютера. Затем оппонент-пользователь вводит свой ход, который так же делается, увеличивая сумму пользователя. Цикл продолжается пока количество оставшихся столбиков больше нуля.

Программа на языке Си

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
#include<graphics.h>

int st[100],max,n,p1sum=0,p2sum=0;

void create(int n){  //Задаём массв st – n столбиков с монетами
for(int i=1;i<=n;i++) st[i]=random(1000);
};


int take(long tn,long a){ //Функция возвращает количество монет в a
int sum=0;                //крайних столбиках, tn – кол-во столбиков
for(int i=tn-a+1;i<=tn;i++) sum+=st[i];
return(sum);
};

long check(long tn, long pt){ //рекурсивная функция
                              //анализа оценок
  long i,j,c=0,c1,c2,lim;
  long res;

 if(tn==0) {res=0; return(res);};  //если столбиков больше нет,
                                   //оценка такому ходу - 0

 res=take(tn,pt); //сколько монет возмёт компьютер этим вариантом?
 tn-=pt;  //убрали взятые столбики
```

```
 if(tn<=1) return(res); //если осталось 1 или 0 столбиков – то оценка
                        //уже есть, возвращаем её
 if(tn<pt) pt=tn;

c=1000000;

for(i=1;i<=pt;i++){  //начинаем перебирать варианты ходов оппонента
  if(tn-i<i) lim=tn-i; else lim=i; //сколько максимально можно взять
                                   //столбиков?
  c1=0;
  for(j=1;j<=lim;j++){ //начинаем перебирать варианты своих ходов
    c2=check(tn-i,j); //получаем оценку хода j
    if(c2>c1) c1=c2; //здесь идёт выбор наибольшей оценки
    };
  if(c1<c) c=c1; //здесь идёт выбор наименьшей из наибольших оценок
  };
res+=c; //выбранная оценка добавляется к взятым монетам…
return(res); //…и всё это возвращается функцией как оценка
};           // исследуемого варианта


void main(){ //выполняется программа отсюда
clrscr();
cin >> n;   //вводим кол-во столбиков
cin >> max; //и макс. первый ход
randomize();
create(n);  //задаём массив столбиков
long p1take,p2take,c,c1,i;

while(n>0){  //главный цикл – пока есть столбики

  c=0;
  cout << "\nWhat do we have:";
  for(i=1;i<=n;i++) cout << " " << st[i]; //вывод имеющихся столбиков
  cout << "\nProcessing...";
  for(i=1;i<=max;i++) //перебираем варианты ходов компьютера
    {c1=check(n,i); //получаем оценку варианта хода в i столбиков
    if(c1>c) {c=c1; p2take=i;}; //выбираем вар. с наибольшей оценкой
    };
  p2sum+=take(n,p2take); //берём столбики в соответствии с
  n-=p2take;             //выбранным вариантом
  max=p2take;
  cout << "\nMy take:" << p2take; //пишем, сколько мы взяли

  if(n==0) break; //если взяли всё – конец игры

  cout << "\nWhat do we have:";
  for(i=1;i<=n;i++) cout << " " << st[i]; //вывод имеющихся столбиков
  cout << "\nYour take?";
  cin >> p1take;  //запрашиваем ход пользователя (оппонента)
  p1sum+=take(n,p1take); //берём для пользователя столько столбиков,
  n-=p1take;             //сколько он запросил
  max=p1take;

  cout << "\nYou have:" << p1sum; //выводим количество монет у
                                  //каждого игрока
  cout << "\nI have: " << p2sum << "\n";
  };
};
```