АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА СЪСТЕЗАТЕЛИ

Решение 1.

За да се определи най-подходящият отбор е необходимо да се разгледат всички възможни варианти. Ако възможните кандидати са n, то различните варианти са всички последователности от 0 и 1 с дължина n. Ако на i-то място в редицата има 1, то i-тия кандидат е включен в отбора, а ако има 0 – не е.

За всеки вариант на отбор се проверява дали състезателите, включени в него познават всички теми. Сред всички отбори, които отговарят на изискването да познават всички теми, се определя този, с най-малък брой състезатели.

Решението се свежда до две задачи – първата е да се генерират всички редици от 0 и 1 с дължина n. Това може да се постигне по различни начини.

Първият начин е да се генерират рекурсивно всички редици. Реализацията на това генериране може да се опише чрез следната функция:

```
void perm(int i)
{
  int j;
  for(j=0;j<=1;j++)
  {
    team[i]=j;
    if(i==n)
    {
    if(checkWorkSet()==m)
    {
      checkMax();
    }
    }
    else perm(i+1);
  }
}
```

Граничният случай отговаря на получаване на поредния отбор и затова в този случай се изпълнява проверка за това дали сформирания за момента отбор покрива всички теми със знанията си. Ако това е изпълнено, се прави проверка за броя състезатели.

Вторият – да се обходят числата от 1 до 2^{10} и да се разгледа тяхното двоично представяне, като се използват побитови операции.

```
for(i=1;i<=(1<<n+1);i++)
  {
   if(checkWorkSet(i)==m)
    {

      checkMax(i);

    }
  }
```

Вторият проблем, който трябва да се разреши, е как да се обединят множествата от теми, които познават състезателите в отбора. Множествата, отговарящи на знанията на всички състезатели се представят с двумерния масив `int sets[16][32]`. Ако i-тия състезател знае j-тата тема, то елемента sets[i][j] има стойност 1, а в противен случай – 0. Обединението на областите на двама състезатели се получава като запишем на j-то място 1 в случай, чу поне един от двамата състезатели знае темата.

Обединението на две множества се реализира с функцията:

```
void unionSet(int k)
{
  int i;
  for(i=0;i<32;i++)
  {
    if(!workSet[i]&&sets[k][i])
    {
      workSet[i]=1;ns++;
    }
  }
}
```

Резултатът се записва в глобалния масив a параметърът показва, кой е поредния номер на състезателя, чиито знания се обединяват с `workSet`.

Накрая функцията

```
int checkWorkSet()
{
  int i;
  initSet();
  ns=0;
  for(i=1;i<=n;i++)
  {

  if(team[i])unionSet(i);
  }
  return ns;
}
```

Обединява областите на знания на всички състезатели и връща като резултат броя им.

Функцията

```
void checkMax()
{
  int i,br=0;
  for(i=1;i<=n;i++)
  if(team[i])br++;
  if(br<=mb)
  {
    mb=br;
    for(i=1;i<=n;i++)maxTeam[i]=team[i];
  }
}
```

Проверява дали броя състезатели в този отбор е по-малък от избрания до момента оптимален отбор и ако е така, обявява новия отбор за оптимален.

Трябва да отбележим, че последните функции се различават при двата различни подхода на решения, тъй като в единият случай отборът се разглежда като масив от 0 и 1, а в другия, като двоично представяне на цяло число.

```
int checkWorkSet(int t)
{
  int i;
  initSet();
  ns=0;
  for(i=1;i<=n;i++)
  {
    if(t&(1<<i))unionSet(i);
```

```
  }
  return ns;
}
void checkMax(int t)
{
  int i,br=0;
  for(i=1;i<=n;i++)
  if(t&(1<<i))br++;
  if(br<mb)
  {
    mb=br;
    maxTeam=t;
  }
}
```

Двата варианта на това решение са реализирани в **competitors.cpp** и **competitors1.cpp.**

Автор: Бисерка Йовчева

Решение 2.

Това решение също използва пълно изчерпване, като идеята е следната. Проверява се дали има един състезател да знае всички теми, започвайки от този с номер 1. Ако има такъв, то първият (с най-малък номер), който се срещне, дава решението на задачата. Ако няма един „универсален" състезател, се преминава към проверка на двойките състезатели, подредени по следния начин: (1,2), (1,3),....,(1,*N*), (2,3), (2,4).....,(2,*N*),.....(*N-1,N*) (казва се, че двойките са подредени в лексикографичен ред). Ако има двойка, която покрива всички теми, то първата срещната в този ред дава решението на задачата. Ако няма, се преминава към тройки и т.н. докато за първи път се срещнат $M \leq N$ състезатели, които покриват всички теми. Първият въпрос, който възниква тук е как да се генерират тези двойки, тройки и т.н. от състезатели в нужния ред (тук може да се каже на децата, че това са комбинации от състезатели, като по този начин се подготвят за заниманията по комбинаторика). При дадено *M,* генерирането на всички *M*- торки в указания по-горе ред може да се извърши с *M* вложени цикъла, т.е. с конструкция от вида:

```
for (i1=1; i1<=(N-M+1); i1++)
  for (i2=i1+1;i2<=(N-M+2; i2++)
    .....................................................
      for(im=i(m-1)+1; im<=N; im++)
```

Между другото, ако някой направи съвсем „хамалско" решение, тръгвайки от един цикъл, след това (ако има нужда) два вложени цикъла, след това (ако има нужда) три вложени цикъла и т.н. най-много до 9 вложени цикъла (да си спомним, че $N \leq 10$), то, при вярна реализация, ще получи 100 точки.

Въпросът е как да се организират *M* вложени цикъла от посочения по-горе вид без да се работи толкова „грубо". За целта трябва да си представим, че когато някой от циклите достигне горната си граница, то предният цикъл „мръдва" с единица нагоре, всички цикли след него инициализират променливите си, като променливата на всеки следващ цикъл приема стойност с единица по-голяма от тази на променливата на предходния и започва отново да се върти най-вътрешният цикъл. Тази техника е реализирана във функция **comb_gen** в **competitors2.cpp.**

В същото решение това кой какви теми знае е запомнено в двумерен масив *t*, като в него редовете съответстват на номерата на състезателите и за всеки ред, в нулевия елемент, е записан броят на темите, които знае съответният състезател, а в следващите(колкото трябва) – номерата на темите, които знае. Използва се булев масив *temi*, в който, при генерирана *M*-торка от състезатели се отбелязват с *true* темите, които тези състезатели знаят (всички елементи на масива *temi* се правят *false* преди проверката на знанията на всяка

новогенерирана *M*-торка от състезатели). Когато всички елементи на масива *temi* станат равни на *true*, то съответната *M*-торка от състезатели е решение на задачата.

Автор: Руско Шиков