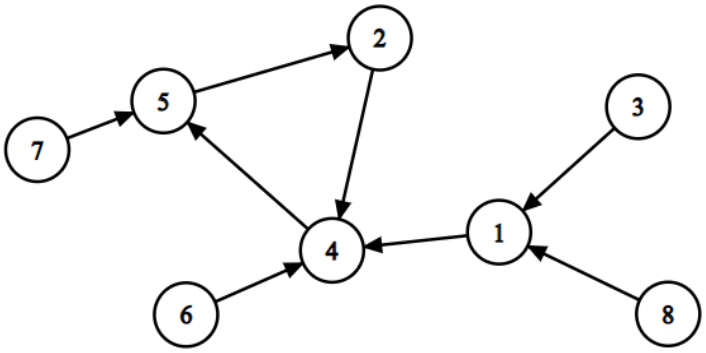
Всероссийская олимпиада школьников по информатике
Вологодская область, 2024-25 учебный год
II (муниципальный) этап
7 - 8 классы
Методические рекомендации по разбору задач

Задача 1. Города без торговых связей

Нарисуем граф и заметим, что он представляет собой цикл, к вершинам которого подвешены ноль, одно или более одного деревьев:



Какие пары вершин не имеют связей? Во-первых, пары вершины из разных деревьев: 1 6, 1 7, 3 6, 3 7, 8 6, 8 7, 6 7

Во-вторых, пары таких вершин в одном дереве, что ни одна из них не является предком другой: 3 8.

В принципе, глядя на рисунок графа, можно было получить ответ, просто перебирая все пары вершин и проверяя, что нельзя попасть ни из первой во вторую, ни наоборот.

Задача 2. Сортировка

Взглянув на последовательность, почти сразу становится ясно, что за один шаг отсортировать её невозможно. Попробуем отсортировать её за два шага.

Заметим, что длина отрезков хотя бы на одном из шагов не может быть меньше 3. Пояснение: у нас 9 элементов не на своих местах, и если бы на обоих шагах длина была бы 2 или меньше, то мы бы не смогли передвинуть все 9 элементов.

Организовать перебор можно так. Вначале предположим, что именно на первом шаге длина отрезков не меньше 3, и будем перебирать, что можно

получить за один шаг из исходной последовательности, используя длины отрезков 3, 4 и 5. Всего получится 15 вариантов с длиной 3, 7 вариантов с длиной 4 и 1 вариант с длиной 5.

Примечание. Если бы мы не нашли ответа, то можно было бы делать то же самое, но в обратную сторону – то есть, идти от последовательности 1 2 3 4 5 6 7 8 9 10. Однако, этого не потребуется, так как ответ найдётся.

Взяв очередной вариант, проверим, что в нём присутствуют ровно два отрезка одинаковой длины из элементов, стоящих не на своих местах. Если это не так, то такой вариант сразу отбросим. Иначе попробуем сделать обмен отрезков и проверим, получилась ли искомая отсортированная последовательность.

На одном из шагов, когда мы пробуем обменять отрезки 3 4 9 10 и 6 7 8 5, получится последовательность 1 6 7 8 5 2 3 4 9 10. В ней ровно два неправильных отрезка 6 7 8 и 2 3 4, и легко увидеть, что при их обмене получится отсортированная последовательность. Таким образом, ответ запишется так:

3 6 4

2 6 3

При желании опытные участники могли делать перебор не вручную, а написав программу. Пример такой программы на Python:

```
a = [1, 3, 4, 9, 10, 2, 6, 7, 8, 5]

def swap(a, i, j, d):
   if i > j:
      i, j = j, i
   return a[:i] + a[j:j+d] + a[i+d:j] + a[i:i+d] + a[j+d:]

def search(a, ans, deep=1):
   for d in range(1, 6):
     for i in range(10):
       for j in range(i + d, 10 - d + 1):
          ans.append((a[i], a[j], d))
          b = swap(a, i, j, d)
          if b == sorted(b):
              for x in ans:
                 print(*x)
              exit(0)
          if deep > 0:
             search(b, ans, deep-1)
          ans.pop()
search(a, [])
```

Используя различные эвристические подходы, участники могли находить и решения с бо́льшим числом шагов. Например, решение за 3 шага, набирающее 80 баллов:
1 2 1
2 6 3
6 1 5
Пример решения за 4 шага, набирающего 60 баллов:
3 2 1
4 7 3
7 6 2
6 3 1

Задача 3. Снова ремонт

Каждый раз мы отрезаем от рулона полосу длины h. Если полоса – не последняя из этого рулона, то затем надо ещё отрезать остаток от обрезанного рисунка, чтобы рулон снова начинался с начала рисунка. То есть, мы отрежем h/k с округлением вверх полных рисунков.

Округлить вверх результат деления можно так: (h+k-1) // k, где операция ‘//’ означает деление нацело. Умножим это на k и получим длину полосы вместе с длиной дополнительного кусочка: segment_len = (h + k - 1) // k * k. Теперь найдём, сколько таких получится из одного рулона: segments = h // segment_len

После этого от рулона останется: restlen = h - segments * segment_len

Заметим, что когда от рулона отрезается последняя полоса, то дополнительный кусок отрезать уже не надо. Поэтому оставшейся длины restlen может хватить ещё на целую полосу – проверим это:

если restlen ≥ h, то segments = segments + 1

Итак, мы нашли, что из одного рулона получается segments полос. Они покроют на стене длину width = segments * m. Тогда, чтобы покрыть всю длину стены, потребуется a / width с округлением вверх рулонов. Округление вверх при делении можно снова записать как (a + width - 1) // width – это и будет ответ.

Пример решения на языке Python:

```
a = int(input())
h = int(input())
k = int(input())
m = int(input())
s = int(input())
segment_len = (h + k - 1) // k * k
segments = s // segment_len
restlen = s - segments * segment_len
if restlen >= h:
```

```
    segments += 1
width = segments * m
rolls = (a + width - 1) // width
print(rolls)
```

В решении на неполный балл можно, было, например, в цикле выполнять отрезание от рулона, пока он не закончится.

Задача 4. Согласование заявок

Заметим, что для согласования всех заявок в любом случае нужно n раз нажать кнопку «Согласовать» после того как отмечены все заявки очередного типа.

Также заметим, что кнопку «Отметить все» не имеет смысла нажимать более одного раза. Если она нажималась хотя бы дважды, то в не последний раз её использования галочки снимались с заявок. Но снятие галочек с заявок не даёт никакого выигрыша: можно было просто за такое же количество нажатий отметить заявки данного типа и согласовать их. Поэтому кнопку «Отметить все» нужно использовать, только когда остались заявки одного типа. И наиболее выгодно, чтобы это был тип с самым большим количеством заявок. Все же остальные заявки будем отмечать по одной. Поэтому ответ на вопрос задачи можно вычислить следующим образом:

$$n + \sum_{i=1}^{n} a_i - \max_{i=1}^{n} a_i + 1$$

Пример решения на языке Python:

```
n = int(input())
a = [0] * n
for i in range(n):
    a[i] = int(input())
print(n + sum(a) - max(a) + 1)
```

Частичные баллы за эту задачу могли получить, например, решения, в которых выполнялась сортировка неэффективным способом, и др.

Задача 5. Треугольники

Первая подзадача решается перебором троек с помощью трёх вложенных циклов и проверкой правила треугольника – каждая сторона меньше суммы двух других. Чтобы не было дубликатов, организуем перебор так, чтобы выполнялось

$a \le b \le c$, то есть цикл по b стартует с a, цикл по c стартует с b: Пример такого решения на Python:

```
P = int(input())
Q = int(input())
ans = 0
for a in range(P, Q + 1):
   for b in range (a, Q + 1):
      for c in range(b, Q + 1):
         if a<b+c and b<a+c and c<a+b:
            ans += 1
print(ans)
```

Чтобы решить вторую подзадачу, устраним самый внутренний цикл. Заметим, что условие $c < a + b$ можно проверять не в теле цикла, а внести его под *range*. Два оставшихся условия в теле цикла избыточны: поскольку у нас $a \le b \le c$, то отсюда верно, что $a < b + c$ и также что $b < a + c$. Получаем такой вариант внутреннего цикла:

```
for c in range(b, min(a + b, Q + 1)):
         ans += 1
```

Отсюда понятно, что вместо прибавления единицы в цикле можно сразу прибавить к ans количество повторений цикла. Получается такое решение второй подзадачи:

```
P = int(input())
Q = int(input())
ans = 0
for a in range(P, Q + 1):
  for b in range (a, Q + 1):
      ans += min(a + b, Q + 1) - b
print(ans)
```

Чтобы решить третью подзадачу, устраним ещё один цикл – теперь по переменной b. Заметим, что в этом цикле функция *min* вначале каждый раз возвращает $a+b$. Затем, начиная с какого-то момента, $a+b$ становится слишком большим, и функция *min* начнёт возвращать $Q + 1$. Это произойдёт в тот момент, когда выполнится равенство $a + b = Q + 1$, то есть когда $b = Q + 1 - a$. Перепишем программу уже с двумя циклами по b, не используя функцию min (это пока всё ещё решение второй подзадачи, только слегка изменённое):

```
P = int(input())
Q = int(input())
ans = 0
for a in range(P, Q + 1):
  for b in range (a, Q + 2 - a):
      ans += a
  for b in range (max(a, Q + 2 - a), Q + 1):
```

```
        ans += Q + 1 - b
print(ans)
```

Теперь заметим, что в первом цикле каждый раз к ответу прибавляется константа, поэтому можно заменить цикл прибавлением к ответу этой константы, умноженной на количество итераций (при подсчёте числа итераций нужно проверить, что оно будет неотрицательным).

Во втором цикле прибавляется константа $Q + 1$ и вычитается переменная b, которая увеличивается на каждом шаге. Сумму этих вычитаний можно найти с помощью формулы суммы арифметической прогрессии, поэтому этот цикл тоже устраняем. Окончательно получается следующее решение на полный балл:

```
P = int(input())
Q = int(input())
ans = 0
for a in range(P, Q + 1):
  if Q + 2 - a > a:
     ans += a * (Q + 2 - a - a)
  if Q + 1 > max(a, Q + 2 - a):
     ans += (Q + 1) * (Q + 1 - max(a, Q + 2 - a))
     ans -= (max(a, Q+2-a) + Q) * (Q + 1 - max(a, Q+2-a)) // 2
print(ans)
```

Примечание. В принципе, можно было попытаться сделать ещё один шаг и устранить цикл по a, получив решение за константное время. Однако, в данной задаче это не требовалось.