Análisis de Algoritmos - Algoritmos básicos en gráficas

Tarea, Teoría de Gráficas (parte 1)

Fecha de entrega: martes 20 de octubre antes de las 23h59 a Francisco Valente

1 Depth-First Search y Topological Sort

Dada una lista de restricciones variables de la forma $A < B$, escribe un programa que imprima todos los ordenamientos posibles de las variables que sea consistente con las restricciones. Por ejemplo, dadas las restricciones $A < B$ y $A < C$, hay dos ordenamientos posibles para las variables A, B y C que sea consistente con las restricciones dadas: ABC y ACB.

Entrada:

La entrada consiste en dos lineas: una lista de variables en una línea, seguida de una lista de restricciones de la forma $A < B$ en la línea siguiente. Las variables y las restricciones están separadas por espacios. Todas las variables son de un solo caracter, en mayúsculas. Hay al menos dos variables y no más de 20 variables. Hay al menos una restricción y no más de 50 restricciones. No habrá más de 300 ordenamientos posibles de acuerdo a la especificación de las variables y restricciones.

Salida:

Se deberán imprimir todos los ordenamientos posibles usando las restricciones. Los ordenamientos estarán escritos en orden alfabético, uno por línea. Los caracteres estarán separados por un espacio. Si no hay ordenamiento posible, la salida es una sola línea con la palabra NO.

Ejemplo de entrada:

```
A B F G
A<B B<F
```

Ejemplo de salida:

```
A B F G
A B G F
A G B F
G A B F
```

2 Breadth-First Search

En una institución de gran tamaño cada persona conoce a muchos de sus colegas pero hay relaciones de amistad solo con un círculo más pequeño, a una persona confía una noticia. Supón que cada vez que un empleado sabe una noticia se la dice a todos sus amigos al día siguiente. Entonces, el primer día, la fuente de información dice a sus amigos; el segundo día, los amigos lo dicen a sus amigos; el tercer día, el amigo del amigo de la fuente lo dice a sus amigos y así sucesivamente. La meta es determinar:

tamaño del máximo boom diario- el número máximo de empleados que, en un solo día, escuchan la noticia por primera vez; y

primer día de boom- el primer día en que ocurre el máximo boom diario.

Escribe un programa que, dadas las relaciones de amistad entre los empleados y el empleado fuente de una noticia, calcule el *número máximo de boom diario* y el *primer día de boom* en el proceso de distribución de la noticia.

Entrada:

La primera línea de la entrada contiene el número E de empleados ($1 \leq E \leq 2500$). Los empleados están numerados de 0 a $E-1$. Cada una de las siguientes E líneas especifica un conjunto de amigos de un empleado (del empleado 0 al empleado $E-1$). Un conjunto de amigos contiene el número de amigos N ($0 \leq N \leq 15$), seguido de N enteros distintos que representa a los amigos. Todos los enteros están separados por un espacio. La siguiente línea contiene un entero $T(1 \leq T \leq 60)$, que es el número de casos de prueba. Cada uno de las siguientes T líneas contiene a un empleado, que representa la fuente (única) de información para la noticia de ese caso de prueba.

Salida:

La salida consiste en T líneas, una para cada caso de prueba. Si ningún empleado (salvo la fuente) escucha la noticia, la salida contiene el entero '0'. Si no, la línea de salida contiene dos enteros, M y D, separados de un espacio, donde M es el tamaño del máximo boom diario y D es el primer día de boom.

Ejemplo de entrada:

```
6
2 1 2
2 3 4
3 0 4 5
1 4
0
2 0 2
3
0
4
5
```

Ejemplo de salida:

```
3 2
0
2 1
```

3 Minimum-Spanning Trees

En el país de Grafoland hay muchas ciudades pero no hay carreteras. El gobierno federal quiere cambiar esta situación y planea construir carreteras y redes de ferrocarriles de tal forma que todas las ciudades del país estén conectadas con este nuevo sistema de transporte. Para hacer las cosas más eficientes, Grafolandia va a construir solo carreteras entre las ciudades del mismo estado y usará las redes de ferrocarriles para conectar las ciudades entre los diferentes estados. Para fines de este problema, considera que si la distancia entre cualquier par de ciudades es a lo más r, entonces están en el mismo estado. Para minimizar los costos de construcción de caminos y rieles de ferrocarril, el gobierno necesita también construir solo la extensión mínima necesaria de éstos, tal que hay un camino entre cualquier pas de ciudades en todo el país. Has sido contratado para determinar cuál es la red de transportación óptima de carreteras y ferrocarriles que Grafolandia tiene que construir.

La primera línea en la entrada contiene el número de casos de prueba que se te proporcionarán. En la primera línea de cada caso de prueba habrá dos enteros, n ($1 \leq n \leq 1000$), el número de ciudades que hay en Grafolandia, y r ($0 \leq r \leq 40000$), el valor umbral para determinar si dos ciudades están en el mismo estado. En las siguientes n líneas, habrá una lista de $x - y$ ($-10000 \leq x, y \leq 10000$) coordenadas enteras en el plan de cada ciudad en Grafoland. Tu programa debe sacar el número de Estados en Grafolandia y su mínima extensión (redondeados al entero más cercano) de ambos, carreteras y vías de ferrocarril que deben construirse para satisfacer las condiciones del proyecto.

Entrada

La primera línea de la entrada da el número de casos, T ($1 \leq T \leq 20$). Siguen los T casos de prueba. En la primera línea de cada caso, habrá dos enteros, n ($1 \leq n \leq 1000$), el número de ciudades que hay en Grafoland, y r ($0 \leq r \leq 40000$), que es el umbral para determinar si dos ciudades están en el mismo estado. En las siguientes n líneas, hay una lista de $x - y$ ($-10000 \leq x, y \leq 10000$) coordenadas enteras en el plan para cada ciudad en Grafoland.

Salida

La salida consta de una línea para cada paso de prueba. La línea identifica al caso de prueba con un número (empezando de uno e incrementando ante cada caso de prueba), seguida de el número de Estados en Grafolandia y la mínima extension (redondeado al entero más cercano) de ambos, carreteras y vías de ferrocarril que se deben construir para satisfacer las condiciones del proyecto.

Nota: Nota que por definición, si A y B están en el mismo estado, y B y C están en el mismo estado, entonces A y C están también en el mismo estado.

Ejemplo de entrada:

```
3
3 100
0 0
1 0
2 0
3 1
0 0
100 0
200 0
4 20
0 0
40 30
30 30
10 10
```

Ejemplo de salida:

```
Case #1:  1 2 0
Case #2:  3 0 200
Case #3:  2 24 28
```

4 Single Source Shortest Paths

Un rascacielos tiene no más de 100 pisos, numerados del 0 al 99. Tiene $n(1 \leq n \leq 5)$ elevadores, los cuáles hacen su recorrido a velocidades distintas. Para cada i en $\{1, 2, \ldots n\}$, el elevador con número i toma T_i $(1 \leq T_i \leq 100)$ segundos para viajar entre dos pisos adyacentes (hacia arriba o hacia abajo). Los elevadores no se detienen necesariamente en todos los pisos, lo que es peor, no todos los pisos son necesariamente accesibles por un elevador.

Tu estás en el piso 0 y quieres llegar al piso k lo más rápido posible. Supón que no necesitas esperar para abordar el primer elevador. La operación de cambiar de elevador en un piso dado toma exactamente un minuto. Claramente, ambos elevadores tienen que detenerse en ese piso. Está prohibido que uses la escalera. No hay ninguna otra persona en el elevador contigo, así que no es necesario hacer escala en otro piso si no quieres. Calcula el número mínimo de segundos requeridos para ir del piso 0 al piso k (pasar por el piso k mientras estás dentro del elevador que no se para allí no cuenta como llegar al piso k).

Entrada:

La entrada consiste en un número de casos de prueba. Cada caso de prueba empieza con dos números n y k en una línea. La siguiente línea contiene los números $T_1, T_2, \ldots, T_n$. Finalmente, las siguientes n líneas contienen las listas ordenadas de enteros – la primera línea lista los pisos visitados por el elevador 1, la siguiente los pisos visitados por el elevador número 2, etc.

Salida:

Para cada caso de prueba, imprimir en una línea un número que indique el número mínimo de segundos que toma llegar al piso k desde el piso 0. Si esto no es posible imprime la palabra `IMPOSSIBLE`.

Explicación de los ejemplos:

- En el primer ejemplo, toma el elevador 1 hasta el piso 13 (130 segundos), espera 60 segundos para cambiar al elevador 2 y viaja en el elevador 2 hasta el piso 30 (85 segundos) para hacer un total de 275 segundos.
- En el segundo ejemplo, toma el elevador 1 hasta el piso 10, cambia al elevador 2 y viaja hasta el piso 25. Allí, regresa al elevador 1 y llega al piso 30. El tiempo total es $10 * 10 + 60 + 15 * 1 + 60 + 5 * 10 = 285$ segundos.
- En el ejemplo 3, toma el elevador 1 hasta el piso 30, cambia al elevador 2 hasta el piso 20 y luego toma el elevador 3 hasta el piso 50.
- En el último ejemplo, no hay elevador que se detenga en el piso 1.

Ejemplo de entrada:

```
2 30
10 5
0 1 3 5 7 9 11 13 15 20 99
4 13 15 19 20 25 30
2 30
10 1
0 5 10 12 14 20 25 30
2 4 6 8 10 12 14 22 25 28 29
3 50
10 50 100
0 10 30 40
0 20 30
0 20 50
1 1
2
0 2 4 6 8 10
```

Ejemplo de salida:

```
275
285
3920
IMPOSSIBLE
```