Informe
Matemática Discreta I
Pashmak and Buses
Problema:459C

Abel Molina Sánchez
Grupo 2-11
Ciencias de la Computación
Universidad de La Habana

26 de diciembre de 2020

1. Problema

C. Pashmak and Buses

Recently Pashmak has been employed in a transportation company. The company has k buses and has a contract with a school which has n students. The school planned to take the students to d different places for d days (each day in one place). Each day the company provides all the buses for the trip. Pashmak has to arrange the students in the buses. He wants to arrange the students in a way that no two students become close friends. In his ridiculous idea, two students will become close friends if and only if they are in the same buses for all d days.
Please help Pashmak with his weird idea. Assume that each bus has an unlimited capacity.

Input:
The first line of input contains three space-separated integers n, k, d ($1 \leq n$, $d \leq 1000$;$1 \leq k \leq 10^9$)

Output
If there is no valid arrangement just print -1. Otherwise print d lines, in each of them print n integers. The *j-th* integer of the *i-th* line shows which bus the *j-th* student has to take on the *i-th* day. You can assume that the buses are numbered from 1 to k.

2. Interpretación e idea

Se tienen n alumnos, d días y k autobuses. Se necesita saber si existe al menos una forma o no de asignar los autobuses a cada estudiante durante los d días de tal forma que no existan 2 estudiantes que coincidan cada día en los mismos autobuses.
Esto es, se necesita conocer si existen al menos n distribuciones distintas de los k autobuses en los d días. En caso de existir hay que generar una de ellas.
Idea:
Conociendo la forma de calcular la cantidad de d-permutaciones de un conjunto de tamaño k, por el principio del palomar se puede conocer si existe una posible solución al problema, ya que en caso de haber menos de n formas distintas de distribución es seguro que al menos 2 alumnos poseerán la misma distribución.
Luego para obtener una de las posibles soluciones se busca generar las primeras n distribuciones de d posiciones donde cada posición va de 1 a k.

3. Algoritmo y correctitud

```
solve(n,k,d):
    if n > k**d:
        fin no solucion
    D = matriz[d][n]
    for i in [0,...,d-1]
        for j in [0,...,n-1]
            D[i,j] = 1
    distribution = [1 for i in [0,..,d-1]]
    for c in [1,....,n-1]
        for j in [d-1,d-2,...,0]
            distribution[j] = (distribution[j]%k)+1
                if distribution[j] != 1
                    break
        for j in [0,...,d-1]
            D[j][c] = distribution[j]
    return D
```

Llamaremos distribución a una fila de d valores, donde cada posición asume valores en $[1, k]$ y que constituye la repartición de los autobuses por días de un alumno. O sea, en cada uno de los d días hay un autobús asignado.
No hay restricciones en cuanto a capacidad de los autobuses, por lo cual todos los alumnos podrían tomar para un día d_i el mismo autobús.
Luego diremos que dos distribuciones, $d1$ y $d2$, son distintas si para una posición i, $d1_i \neq d2_i$. O sea, si no tienen igual valor en todas las posiciones.
Diremos que una distribución $d1$ es mayor que una distribución $d2$ si en el recorrido i=[0,..,d-1], en la primera posición donde $d1_i \neq d2_i$, se tiene que $d1_i > d2_i$.

Una vez vistas las definiciones , tenemos que buscar una distribución distinta para cada alumno. Tenemos que el número de posibles distribuciones será igual a k^d que es el número de d-permutaciones con repetición de un conjunto de tamaño k (cada posición d_i tiene k opciones de autobuses, 1 entre $\{1, 2, ..., k\}$, con lo cual $k * k... * k = k^d$).
Luego habrá solución si logramos que exista una distribución distinta para cada alumno, luego como hay n alumnos y k^d distribuciones, por el Principio del palomar sabemos que para tener una solución es necesario que $n \leq k^d$ ya que si hay más alumnos que posibles distribuciones, al menos 2 alumnos tendrán la misma distribución, con lo cual no habría solución.
Luego en la condición *if* inicial se comprueba la existencia o no de solución.
Ahora necesitamos ver que las n distribuciones que se crean en el ciclo son válidas y distintas.
Que son válidas se garantiza porque tenemos que los restos con k son $[0, 1, ..., k-1]$, y por tanto los restos $+1$ estarán en el rango $[1, ..., k]$ y todos se alcanzan porque sabemos que $i \in [1, ..., k-1]$ deja resto i con k y $k \,\%k = 0$.
Luego para demostrar que las distribuciones formadas son distintas nos apoyaremos en las comparaciones definidas anteriormente, demostrando que cada distribución es mayor que la anterior.
Vamos a hacerlo por inducción en la iteración del donde se forma la distribución.
Tenemos que en un primer momento se tiene la distribución $d1 = [d_i = 1 \ \forall i \in [0, .., d-1]]$ que es

una posible. Luego la primera que se forma en el ciclo, es la distribución
$d2 = [d_0 = 1, d_1 = 1, ..., d_{d-2} = 1, d_{d-1} = 2]$, que es mayor que $d1$.
Ahora asumamos por hipótesis que se cumple para la distribución dk.
Sea ahora el momento donde se está formando la distribución $dk + 1$, tenemos que en ese momento *distribution* es igual a dk. Luego, como se recorren las posiciones desde $[d - 1, .., 0]$, sea j la primera posición donde $distribution[j] \neq 1$. Tenemos que $distribution_{i+1}[j]$ se hace 1 solo cuando $distribution_i[j] = 5$, porque en $[1, ..., k]$ que son los valores alcanzables solo $k\,\%k = 0$ y por tanto $k\,\%k + 1 = 1$.
Luego tenemos que para j $dk[j] = i \in [1, ..., k - 1]$, por tanto $i\,\%k = i$ e $i + 1 > i$, con lo cual la distrubución $dk + 1$ es mayor que la distribución de k. Luego queda demostrado que cada distribución es mayor que la anterior y por tanto distintas.

Complejidad temporal

Determinar si se cumple o no el principio del palomar es $O(d)$.
Luego para generar las n distribuciones de los estudiantes se realizan dos ciclos for anidados, el primero realiza $O(n)$ iteraciones y el segundo $O(d)$ iteraciones por lo cual completar las distribuciones es $O(n * d)$.
Luego para imprimir la solucion se recorre todas las distribuciones, con lo cual se realiza nuevamente en $O(n * d)$, con lo cual por regla de la suma se tiene que la complejidad temporal de la solución es $O(n * d)$.

4. Tester

Como parte de la realización del ejercicio se incluye un tester y un generador de casos. Ambos estan incluidos en la carpeta *tester-casos-prueba* junto con una copia de la solución.
Dentro del generador se incluyen dos métodos para generar una entrada para el problema. Uno crea el archivo *casoX.in* con los parametros elegidos de forma manual, y el otro genera la entreda generando 3 enteros de forma aleatoria utilizando el método *randint(a,b)* de la libreria random de python.
El *tester.py* para comprobar las soluciones de cada caso de prueba:
Lee la entrada y comprueba por el Principio del polomar que el caso deba tener o no solución, en caso de no tenerla comprueba que la salida del algoritmo haya sido '-1'.
Luego lee la salida completa del algoritmo y comprueba que haya sido con el formato correcto, esto es: d filas con n columnas.
Se crea un *set* para almacenar las distribuciones de cada estudiante y comprobar que no hay 2 iguales. Se recorre la matriz de solución comprobando que cada número en cada distribución está entre 1 y k(representan un autobús valido) y se van agregando las distribuciones al set, si se detectan 2 iguales en algún momento se determina una solución errónea, en caso contrario luego de terminada la comprobación la solución sería correcta.
El tester comprobará todos los tests incluidos en la carpeta *./tests* con un '.out'. Por tanto es importante que si se agrega algún nuevo caso para trabajar la solución este tenga un '.in' y un '.out' con igual nombre y diferente al del resto de casos dentro de la carpeta *tester*.
Como casos de pruebas que se consideran importantes analizar están aquellos que puedan co-

siderarse extremos dentro de la ejecución. Comprobamos para cuando $n = k = d = 1$(caso1), comprobamos el test de estrés del algoritmo para cuando cada parámetro recibe el máximo valor posible dentro de las restricciones del problema, esto es $n = d = 1000$ y $k = 10^9$(caso3). Se comprueba para cuando no existe solución y deja de existir por una diferencia mínima, esto es, cuando $n = k^d + 1$, lo hacemos con $n = 65$, $k = 4$, $d = 3$(caso4). Comprobamos cuando $n = k^d$, lo hacemos con $n = 64(27)$, $k = 4(3)$, $d = 3(3)$(casos 5 y (3)).