DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Diseño de Algoritmos

Aplicación de técnicas 2.0

Algoritmos y Estructuras de Datos III
Primer Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Barañao, Facundo	480/11	facundo_732@hotmail.com
Confalonieri, Gisela Belén	511/11	gise_5291@yahoo.com.ar
Mignanelli, Alejandro Rubén	609/11	minga_titere@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Objetivos generales

2. Plataforma de pruebas

Para toda la experimentación se utilizará un procesador Intel Atom, de 4 núcleos a 1.6 GHZ.
El software utilizado será Ubuntu 14.04, y G++ 4.8.2.

3. Problema 1: Dakkar

3.1. Descripción del problema.

Tenemos la intención de competir en una versión particular del Rally Dakkar, y para esto decidimos usar nuestros conocimientos en computación a nuestro a favor. Sabemos que en esta competencia podremos usar una BMX, una motocross y un buggy arenero. Nuestro objetivo es finalizar la competencia en el menor tiempo posible, y contamos con los siguientes datos:

- El circuito se divide en n etapas (numeradas del 1 al n) y en cada etapa sólo se puede usar un vehículo.
- Para cada etapa, se sabe cuánto se tardaría en recorrerla con cada vehículo.
- Tanto la motocross como el buggy arenero tienen un número limitado de veces que se pueden usar, a diferencia de la BMX, y estos números son conocidos.
- La complejidad del algoritmo pedido es de $\mathcal{O}(n \cdot k_m \cdot k_n)$ donde n es la cantidad de etapas, k_m es la cantidad máxima de veces que puedo usar la moto, y k_b la cantidad máxima de veces que puedo usar el buggy.
- La salida de este algoritmo deberá mostrar el tiempo total de finalización de la carrera y una sucesión de n enteros representando el vehículo utilizado en cada etapa (1 para bici, 2 para moto y 3 para buggy).

Ejemplo:
Supongamos un Rally de 3 etapas, en donde puedo usar una vez la moto y una vez el buggy, con los siguientes datos:

Etapa	BMX	Moto	Buggy
1	10	8	7
2	8	3	7
3	15	6	2

Con estos datos, la manera óptima de llevar a cabo la carrera sería la siguiente:

Etapa 1	Etapa 2	Etapa 3	Tiempo total
BMX: 10	Moto: 3	Buggy: 2	15

3.2. Desarrollo de la idea y correctitud.

LLamemos n a la cantidad de etapas del Rally, k_m y k_b a la cantidad máxima de motos y buggys respectivamente que es posible utilizar en el Rally. La idea es no tratar de resolver todas las etapas de un tirón, sino tratar primero de resolver considerando sólo la primer etapa, luego considerando las dos primeras etapas ayudándonos con la resolución de la primera etapa sola, luego considerando las primeras tres etapas ayudándonos con la resolución de las primeras dos etapas, y así hasta llegar a considerar n etapas. Para esto, la resolución que considera las primeras h etapas tendrá una tabla construida a partir de la información de la tabla correspondiente a la resolución de las primeras $h-1$ etapas. Cada una de estas tablas tiene $k_m + 1$ columnas (numeradas del 0 al k_m) y $k_b + 1$ filas (numeradas del 0 al k_b). Para hacer referencia a la tabla que representa a las primeras t etapas (con t algún número natural distinto de 0), la llamaremos la tabla t. Entonces si tomamos $i, j, h \in \mathbb{N}$ tal que $0 \leq i \leq k_b$, $0 \leq j \leq k_m$ y $1 \leq h \leq n$

la idea sería que la posición i, j de la tabla h tenga dos datos: el tiempo óptimo de las primeras h etapas usando a lo sumo i buggies y j motos, y el vehículo que se usaría en la etapa h bajo esas circunstancias.

Determinemos ahora cómo se completarían las tablas (para hacer referencia a una posición de una tabla h, diremos la posición $h_{f,c}$ donde f es el número de fila, y c el número de columna):

- Si $h = 1$
  - En la posición $h_{0,0}$ colocaremos el tiempo de la bicicleta en la etapa h, puesto que sólo puede usarse ese vehículo, e indicaremos que se usó la bici.
  - En la posición $h_{0,1}$ colocaremos el tiempo de la bicicleta o la moto (ambos en la etapa h), el que sea menor, e indicaremos cuál de ellos se utilizó.
  - De la posición $h_{0,2}$ hasta h_{0,k_m} colocaremos exactamente lo que hay en $h_{0,1}$ dado que si consideramos una sola etapa, poder usar una moto o más de una, no es diferencia.
  - En la posición $h_{1,0}$ colocaremos el tiempo de la bicicleta o el buggy (ambos de la etapa h), el que sea menor, e indicaremos cuál de ellos se utilizó.
  - De la posición $h_{2,0}$ hasta $h_{k_b,0}$ colocaremos exactamente lo que hay en $h_{1,0}$ dado que si consideramos una sola etapa, poder usar un buggy o más de uno, no es diferencia.
  - En la posición $h_{1,1}$ colocaremos el tiempo de la bicicleta, el buggy o la moto (de la etapa h), el que sea menor, e indicaremos cuál de ellos se utilizó.
  - El resto de las posiciones de esa tabla, deben tener lo que hay en $h_{1,1}$ dado que si tengo una sola etapa, poder usar una moto y un buggy, o más de alguno de ellos o de ambos, no es diferencia.
- Para todo h tal que $1 < h \leq n$
  - En la posición $h_{0,0}$ colocaremos el tiempo que de la posición $(h-1)_{0,0}$ sumada al de la bici de la etapa h, e indicaremos que se usó la bici. El tiempo óptimo de este caso es correcto, puesto que al no poder usar otros vehículos, el tiempo óptimo de haber recorrido h etapas usando solo la bicicleta, es el tiempo óptimo de haber recorrido $h-1$ etapas sin usar buggys ni motos, sumado a usar una bicicleta en la etapa h. El vehículo elegido es trivialmente correcto, puesto que no hay otra elección posible.
  - Tomando $1 \leq j \leq k_m$, la posición $h_{0,j}$ debe contener el tiempo mínimo entre:
    1. El tiempo de $(h-1)_{0,j}$ sumado al tiempo de la bici en la etapa h.
    2. El tiempo de $(h-1)_{0,(j-1)}$ sumado al tiempo de la moto en la etapa h.

    Para el primer caso indicaremos que se eligió la bici, y para el segundo caso indicaremos que se eligió la moto. Veamos que el tiempo escogido es óptimo:
    1. Si el vehículo que deberíamos tomar en la etapa h fuera la bicicleta, entonces en las primeras $h-1$ etapas pudimos haber usado hasta j motos (dado que no vamos a usar ninguna moto en esta etapa). Como $(h-1)_{0,j}$ indica el tiempo óptimo de usar hasta j motos en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar la bicicleta en la etapa h obtenemos el tiempo óptimo para esta etapa.
    2. Si el vehículo que deberíamos tomar en la etapa h fuera la moto, entonces en las primeras $h-1$ etapas pudimos haber usado hasta $j-1$ motos (dado que vamos a usar una moto en esta etapa). Como $(h-1)_{0,(j-1)}$ indica el tiempo óptimo de usar hasta $j-1$ motos en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar la moto en la etapa h obtenemos el tiempo óptimo para esta etapa.

    Como tomamos el mínimo entre 1) y 2) entonces, y ambos, de ser la respuesta correcta, serían óptimos, entonces la elección resulta óptima en tiempo. Por esto mismo, el vehículo elegido es el correcto. Ver que no se crean conflictos con el buggy, ni tenemos que considerar elegirlo o no, puesto que no podemos usar buggies en estos casos.
  - Tomando $1 \leq i \leq k_b$ la posición $h_{i,0}$ debe contener el tiempo mínimo entre:
    1. El tiempo de $(h-1)_{i,0}$ sumado al tiempo de la bici en la etapa h.

2. El tiempo de $(h-1)_{(i-1),0}$ sumado al tiempo del buggy en la etapa h.

Para el primer caso indicaremos que se eligió la bici, y para el segundo caso indicaremos que se eligió el buggy. Veamos que el tiempo escogido es óptimo:

1. Si el vehículo que deberíamos tomar en la etapa h fuera la bicicleta, entonces en las primeras $h-1$ etapas pudimos haber usado hasta i buggies (dado que no vamos a usar ningún buggy en esta etapa). Como $(h-1)_{i,0}$ indica el tiempo óptimo de usar hasta i buggies en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar la bicicleta en la etapa h obtenemos el tiempo óptimo para esta etapa.
2. Si el vehículo que deberíamos tomar en la etapa h fuera el buggy, entonces en las primeras $h-1$ etapas pudimos haber usado hasta $i-1$ buggies (dado que vamos a usar un buggy en esta etapa). Como $(h-1)_{(i-1),0}$ indica el tiempo óptimo de usar hasta $i-1$ buggies en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar un buggy en la etapa h obtenemos el tiempo óptimo para esta etapa.

Como tomamos el mínimo entre 1) y 2) entonces, y ambos, de ser la respuesta correcta, serían óptimos, entonces la elección resulta óptima en tiempo. Por esto mismo, el vehículo elegido es el correcto. Ver que no se crean conflictos con la moto, ni tenemos que considerar elegirla o no, puesto que no podemos usar motos en estos casos.

- Tomando $1 \leq i \leq k_b$ y $1 \leq j \leq k_m$ la posición $h_{i,j}$ debe contener el tiempo mínimo entre:
  1. El tiempo de $(h-1)_{i,j}$ sumado al tiempo de la bici en la etapa h.
  2. El tiempo de $(h-1)_{i,(j-1)}$ sumado al tiempo de la moto en la etapa h.
  3. El tiempo de $(h-1)_{(i-1),j}$ sumado al tiempo del buggy en la etapa h.

  Para el primer caso indicaremos que se eligió la bici, para el segundo caso indicaremos que se eligió la moto y para el tercer caso indicaremos que se eligió el buggy. Veamos que el tiempo escogido es óptimo:

  1. Si el vehículo que deberíamos tomar en la etapa h fuera la bicicleta, entonces en las primeras $h-1$ etapas pudimos haber usado hasta i buggies y j motos (dado que no vamos a usar ningún buggy ni moto en esta etapa). Como $(h-1)_{i,j}$ indica el tiempo óptimo de usar hasta i buggies y hasta j motos en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar la bicicleta en la etapa h obtenemos el tiempo óptimo para esta etapa.
  2. Si el vehículo que deberíamos tomar en la etapa h fuera la moto, entonces en las primeras $h-1$ etapas pudimos haber usado hasta $j-1$ motos y hasta i buggies (dado que vamos a usar una moto en esta etapa). Como $(h-1)_{i,(j-1)}$ indica el tiempo óptimo de usar hasta i buggies y hasta $j-1$ motos en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar la moto en la etapa h obtenemos el tiempo óptimo para esta etapa.
  3. Si el vehículo que deberíamos tomar en la etapa h fuera el buggy, entonces en las primeras $h-1$ etapas pudimos haber usado hasta $i-1$ buggies y hasta j motos (dado que vamos a usar un buggy en esta etapa). Como $(h-1)_{(i-1),j}$ indica el tiempo óptimo de usar hasta $i-1$ buggies y hasta j motos en las primeras $h-1$ etapas, al sumarle el tiempo de utilizar un buggy en la etapa h obtenemos el tiempo óptimo para esta etapa.

  Como tomamos el mínimo entre 1), 2) y 3), y todos, de ser la respuesta correcta, serían óptimos, entonces la elección resulta óptima en tiempo. Por esto mismo, el vehículo elegido es el correcto.

Entonces, una vez completas las tablas, si queremos saber cuál es el menor tiempo en el que podemos completar el Rally, tan sólo debemos ver el tiempo de n_{k_b,k_m}, y si queremos saber qué vehículos utilizamos en cada etapa, debemos ver qué vehículo se usó en n_{k_b,k_m} y ese vehículo será el que se usó en la etapa n. Luego, de manera general, suponiendo $2 \leq h \leq n$, para extraer el vehículo utilizado en la etapa $h-1$, habiendo extraído el vehículo de la etapa h de la posición $h_{i,j}$ con $0 \leq i \leq k_b$ y $0 \leq j \leq k_m$, por cómo está hecha la tabla, debemos ir a:

- la posición $(h-1)_{i,j}$ si en la etapa h elegí la bici.

- la posición $(h-1)_{i,j-1}$ si en la etapa h elegí la moto.
- la posición $(h-1)_{i-1,j}$ si en la etapa h elegí el buggy.

Y en la posición a la que haya ido, extraer el vehículo utilizado, que será el correspondiente a la etapa $h-1$. Haciendo esto en orden, se obtendría qué vehículo fue utilizado en cada etapa. De esta manera, se resolvería el problema de manera correcta.

3.3. Análisis de complejidad.

```
DAKKAR(n, km, kb, bicis, motos, buggies)
 1  etapa_1(0,0) ← [bicis_1,bici]
 2  for j = 1 to km
 3       etapa_1(0,j) ← ELECCIÓN(bicis_1,motos_1)
 4  for i = 1 to kb
 5       etapa_1(i,0) ← ELECCIÓN(bicis_1,buggies_1)
 6  for i = 1 to kb
 7       for j = 1 to km
 8            etapa_1(i,j) ← ELECCIÓN(bicis_1,motos_1,buggies_1)
 9  for h = 1 to n
10       etapa_h(0,0) ← [etapa_{h-1}(0,0)+bicis_h, bici]
11       for j = 1 to km
12            etapa_h(0,j) ← ELECCIÓN(etapa_{h-1}(0,j) + bicis_h,etapa_{h-1}(0,j-1) + motos_h)
13       for i = 1 to kb
14            etapa_h(i,0) ← ELECCIÓN(etapa_{h-1}(i,0) + bicis_h,etapa_{h-1}(i-1,0) + buggies_h)
15       for i = 1 to kb
16            for j = 1 to km
17                 etapa_h(i,j) ← ELECCIÓN(etapa_{h-1}(i,j) + bicis_h,etapa_{h-1}(i,j-1) + motos_h,
                                                                  etapa_{h-1}(i-1,j) + buggies_h)
```

Figura 1: Pseudocódigo del algoritmo para Dakkar

```
DAKKAR_SALIDA(etapas, km, kb)
1  i ← kb
2  j ← km
3  mostrar tiempo_total ← etapa_n(i,j).tiempo
4  for h = n to 1
5       mostrar vehículo ← etapa_h(i,j).vehiculo
6       if vehículo == moto
7            decrementar j
8       else if vehículo == buggy
9                 decrementar i
```

Figura 2: Pseudocódigo del armado de la salida de Dakkar

Para empezar a analizar la complejidad de nuestro algoritmo veamos brevemente como funciona la función ELECCIÓN. Esta, por medio de una serie de simples comparaciones entre los tiempos de la bicicleta, la moto y el buggy $\mathcal{O}(1)$, devuelve el valor y el vehículo que, para la etapa en cuestión, resultan

óptimos.
Partiendo de esta base, observemos que nuestro algoritmo se encarga de rellenar n matrices de tamaño $km * kb$, haciendo uso de ELECCIÓN, cuya complejidad ya dijimos, es $\mathcal{O}(1)$. En consecuencia este proceso posee una complejidad de $\mathcal{O}(n * km * kb)$.
Hasta aquí hemos cubierto la función DAKKAR, la cual como veremos en breve, es la que en definitiva termina asginando la complejidad total del algoritmo planteado. Esto se debe a que, una vez completadas las n matrices, lasolución se arma recorriendo, dependiendo de la elección del vehículo, una casilla determinada de cada matriz "etapa", agregando el tiempo incurrido en la etapa en la lista solución y acumulando los tiempos para obtener el tiempo total utilizado. Como tenemos n matrices y tanto acceder a una posición de una matriz como acumular los tiempos y armar la lista solución es $\mathcal{O}(1)$, este proceso toma $\mathcal{O}(n)$.
Por útlimo, mostrar el resultado consiste en recorrer una lista de n elementos y mostrar cada uno de estos en $\mathcal{O}(1)$
Para concluir, sea $T(n)$ la complejidad de nuestro algoritmo, tenemos:

$$T(n) = \mathcal{O}(n * km * kb) + 2\mathcal{O}(n)$$
$$T(n) = \mathcal{O}(n * km * kb)$$

3.4. Experimentación y gráficos.

3.4.1. Test 1

3.4.2. Test 2

4. Problema 2: Zombieland II

4.1. Descripción del problema.

Luego del éxito rotundo del último ataque contra los zombies producido por nuestro anterior algoritmo, la humanidad ve un rayo de esperanza.

En una de las últimas ciudades tomadas por la amenaza Z, se encuentra un científico que afirma haber encontrado la cura contra el zombieismo, rodeado por una cuadrilla de soldados del más alto nivel. Nuestro noble objetivo es, entonces, proveer del camino más seguro (el que genere menos perdidas humanas) al científico y sus soldados de manera tal que logren llegar desde donde están, hasta el bunker militar que se encuentra en esa ciudad. Para esto, se conocen los siguientes datos:

- La ciudad en cuestión tiene la forma clásica de grilla rectangular, compuesta por n calles paralelas en forma horizontal, m calles paralelas en forma vertical dando así una grilla de manzanas cuadradas. Los numeros n y m son conocidos.
- Se conoce la ubicación del científico y del bunker.
- Se conoce la cantidad de soldados que el científico tiene a su disposición.
- Si todos los soldados perecen, el científico no tiene posibilidad de sobrevivir por sí mismo (o sea, no podemos llegar al bunker con 0 soldados).
- Se sabe la cantidad de zombies ubicados en cada calle.
- Si bien nuestros soldados tienen un alto nivel de combate, el enfrentamiento que tuvieron en el último ataque contra los zombies, acabaron con todas sus municiones por lo cual solo tienen sus cuchillos de combate. Esto incide entonces, en que el grupo sólo pasará por una calle sin sufrir pérdidas si hay hasta un soldado por zombie, al menos, y en caso contrario, perderá la diferencia entre la cantidad de soldados, y la cantidad de zombies. En otras palabras, sea z la cantidad de zombies de una calle, y s la cantidad de soldados de que tiene la cuadrilla al momento de pasar por esa calle, si $s \geq z$ entonces la cuadrilla pasa sin perdidas; en caso contrario la cuadrilla pierde $z - s$ soldados.
- Puede no existir un camino que asegure la supervivencia del científico.
- La complejidad del algoritmo pedido es de $\mathcal{O}(s \cdot n \cdot m)$.
- La salida de este algoritmo deberá mostrar la cantidad de soldados que llegan vivos al bunker, seguido de una línea por cada esquina del camino recorrido, representada por dos enteros indicando la calle horizontal y la vertical que conforman dicha intersección. En caso de no existir solución, se mostrará el valor 0.

Ejemplo:

Supongamos una ciudad como muestra la Figura 3, donde los números indican la cantidad de zombies en cada cuadra. Consideremos que la cantidad de soldados al comenzar es 10. La solución para este ejemplo permite llegar al búnker sin perder ningún soldado, siendo el recorrido el indicado en la Figura 4.

4.2. Desarrollo de la idea y correctitud.

Debemos recorrer la ciudad en busca de un camino que permita llegar al búnker con la mayor cantidad de soldados vivos. Sin embargo, probar todos los posibles caminos tomaría más tiempo del que disponemos en un ataque zombie. Por lo tanto, hemos decidido buscar dicho camino de la siguiente manera:

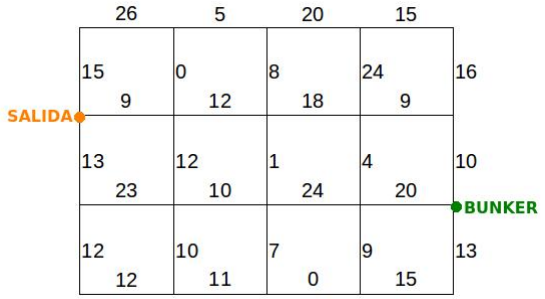

Figura 3: Ejemplo de ciudad para Zombieland II

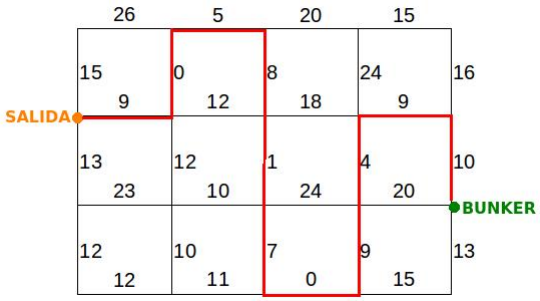

Figura 4: Solución para el problema de la Figura 3

Utilizando *backtraking*, trataremos de encontrar las rutas que salen desde la posición de origen, de manera que ningún soldado muera. Cada cuadra visitada será marcada para que ningún otro camino intente pasar por ahí, y así no tendremos caminos redundantes (es decir, no consideraremos distintos caminos que permitan llegar a una misma esquina con la misma cantidad de soldados vivos), y para cada esquina alcanzada se guardará la esquina anterior de ese camino y la cantidad de soldados vivos hasta ese momento. Si en algún momento se llega a una esquina con alguna cuadra incidente por la cual no se puede avanzar sin pérdidas, vamos a señalizarla (en breve explicaremos para qué).

Si por alguno de estos caminos se llega al búnker, entonces habremos encontrado una solución óptima. Si ninguna de las rutas encontradas es solución, entonces deberemos animarnos a perder soldados. Para ello, primero vamos a arriesgar sólo un soldado, es decir, si teníamos s soldados iniciales, intentaremos llegar al búnker con $s-1$ soldados vivos. Entonces, retomaremos los caminos marcados anteriormente, a partir de las esquinas desde las que no podíamos avanzar sin pérdidas (reanudando cada una en el orden en el que fueron marcadas). Desde ahí, avanzaremos armando los caminos que no nos produzcan más de una baja, y lo haremos de la misma manera que explicamos arriba: guardando la esquina antecesora y la cantidad de soldados vivos en cada esquina atravesada, y sin considerar distintos caminos que permitan llegar a una misma esquina con la misma cantidad de soldados vivos, descartando además los caminos que, habiendo perdido un soldado, llegan a una cuadra atravesada anteriormente por otro camino con más soldados vivos. Si al intentar retomar el camino desde una esquina marcada anteriormente, una de las cuadras incidentes produce más de una baja en los soldados, entonces la esquina seguirá estando marcada para alguna etapa posterior.

Nuevamente, si alguno de estos caminos llega al búnker, será la solución. Caso contrario repetiremos el procedimiento intentando llegar con $s-2$ soldados vivos. De forma sucesiva, nos animaremos a perder cada vez un soldado más, y el primer camino encontrado que llegue al búnker será retornado. De esta manera, estamos asegurando que la solución tiene la mayor cantidad de soldados vivos al final.

Podría suceder que, durante el recorrido, se llegue a una esquina a la cual incida alguna cuadra que elimine la totalidad de soldados que se encuentran vivos hasta ese momento. Esto quiere decir que, por esa cuadra, no existe ningún camino posible hasta el búnker. Pero por como hemos diseñado el algoritmo, esa esquina quedaría marcada y cada vez que arriesguemos un soldado intentaremos avanzar inútilmente. Para evitar estas vanas consultas, decidimos que en estos casos la cuadra deje de ser considerada. En consecuencia, si a partir de cierto momento todas las cuadras que puedo tomar tienen una cantidad de zombies suficiente como para vencer a todos los soldados vivos, entonces no existe una solución.

4.3. Análisis de complejidad.

4.4. Experimentación y gráficos.

4.4.1. Test 1

4.4.2. Test 2

5. Problema 3: Refinando petróleo

5.1. Descripción del problema.

Tenemos en cierta zona, pozos de petróleo que necesita ser refinado. Para que un pozo pueda refinar su petróleo, necesita o bien una refinería ubicada junto a él, o bien conectarse mediante tuberías a otro pozo que o tenga una refinería, o esté conectado mediante tuberías a algun pozo que puede refinar su petróleo. Nuestro objetivo es armar un plan de construcción que decida dónde construir una refinería y dónde construir un sistema de tuberías, de manera tal que todos los pozos puedan refinar su petróleo, minimizando el costo. Para esto, se tienen los siguientes datos:

- Se conoce la cantidad de pozos en cuestión.
- Se conoce el costo de construir una refinería (este costo es fijo, y no varía según el pozo).
- Dada la geografía del lugar, no todo par de pozos se puede comunicar por una tubería directamente, y por decisiones de administración, una tubería no puede bifurcarse a mitad de camino entre un pozo y otro. Igualmente, conocemos todos los pares de pozos que pueden conectarse con una tubería, y el costo de ésta en caso de decidir construirse (a diferencia de las refinerías, las tuberías dependen del par de pozos que se quiere conectar).
- La complejidad del algoritmo debe ser estrictamente menor a $\mathcal{O}(n^3)$.
- La salida de este algoritmo debe contener una línea con el costo total de la solución, la cantidad de refinerías y la cantidad de tuberías a construir, seguido de una línea con los números de pozos en los que se construirán refinerías, más una línea con dos números por cada tubería a construir, representando el par de pozos conectados.

Ejemplo:
PLANTEAR EJEMPLO CON UN DIBUJITO, ES MÁS ENTENDIBLE... SI, ES IGUAL QUE ZOMBIELAND, PERO TODO LO QUE TENGA QUE VER CON GRAFOS LO QUEREMOS VER EN UN DIBUJITO..(ALE)

5.2. Desarrollo de la idea y correctitud.

5.3. Análisis de complejidad.

5.4. Experimentación y gráficos.

5.4.1. Test 1

5.4.2. Test 2

5.4.3. Test 3

6. Apéndice 1: acerca de los tests

7. Apéndice 2: secciones relevantes del código

En esta sección, adjuntamos parte del código correspondiente a la resolución de cada problema que consideramos más **relevante**.

7.1. Código del Problema 1

7.2. Código del Problema 2

7.3. Código del Problema 3