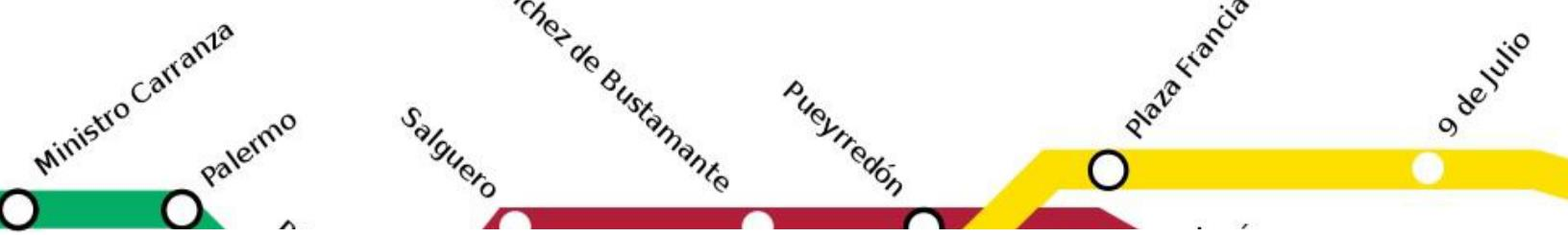


Análisis y Diseño de Algoritmos II

Trabajo Practico Especial

Ingeniería de sistemas

Universidad Nacional d
Centro de Buenos Aire
Facultad de Ciencias Exactas

Arispe Agustín, Pianzola Luca
Profesor a Cargo: Federico Amendola

Índice

Introducción

Durante el desarrollo de todo este informe, se verán detallados todas las partes que componen al programa pedido, un análisis completo del entorno con un ejemplo de ejecución, y una estimación de la complejidad temporal del algoritmo.

Por cuestiones de facilidad e implementación, decidimos utilizar funciones fuera de lo pedido por el profesor, lo cual nos facilitó el trabajo.

Objetivo

El objetivo de esta segunda parte del trabajo será resolver el problema planteado mediante dos técnicas algorítmicas distintas: Backtracking y Greedy.

Luego se deberán comparar los resultados teniendo en cuenta distintas métricas que permitan visualizar, mínimamente, la calidad de la solución y el costo de obtener dicha solución, con ambas técnicas.

Consigna

Las autoridades de una ciudad deciden construir una red de subterráneos para resolver los constantes problemas de tráfico. La ciudad ya cuenta con N estaciones construidas, pero todavía no tienen ningún túnel que conecte ningún par de estaciones entre sí.

La red de subterráneos que se construya debe incluir a todas las estaciones (es decir, que de cualquier estación H pueda llegar a cualquier otra estación J, ya sea de manera directa o atravesando otras estaciones).

Sin embargo, debido al acotado presupuesto, las autoridades desean construir la menor cantidad de metros de túnel posibles. Para esto han calculado cuantos metros de túnel serían necesarios para conectar de manera directa cada par de estaciones existentes.

Representación

Decidimos utilizar como estructura a representar un grafo con costos asociados no dirigidos, donde cada vértice corresponde a cada estación "Ei" (0<i<=N), y adyaciendo de él, sus respectivos túneles a realizar, representados con arcos de costo establecido. He aquí un ejemplo de la representación (figura 1).

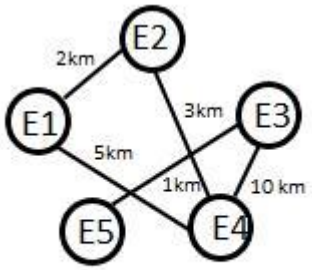


figura 1

Implementación

Aclaración: el cálculo de complejidad es estimativo y basado en la teoría explicada en clase, el costo real del algoritmo está calculado en el proyecto.

Greedy

En este caso utilizamos el algoritmo de Prim, El algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible, Ejemplo (figura 2). Si el grafo no es conexo, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicho grafo no conexo. Consta de una complejidad O(n^2) por lo que es bastante menos costoso que Backtracking (a continuación).

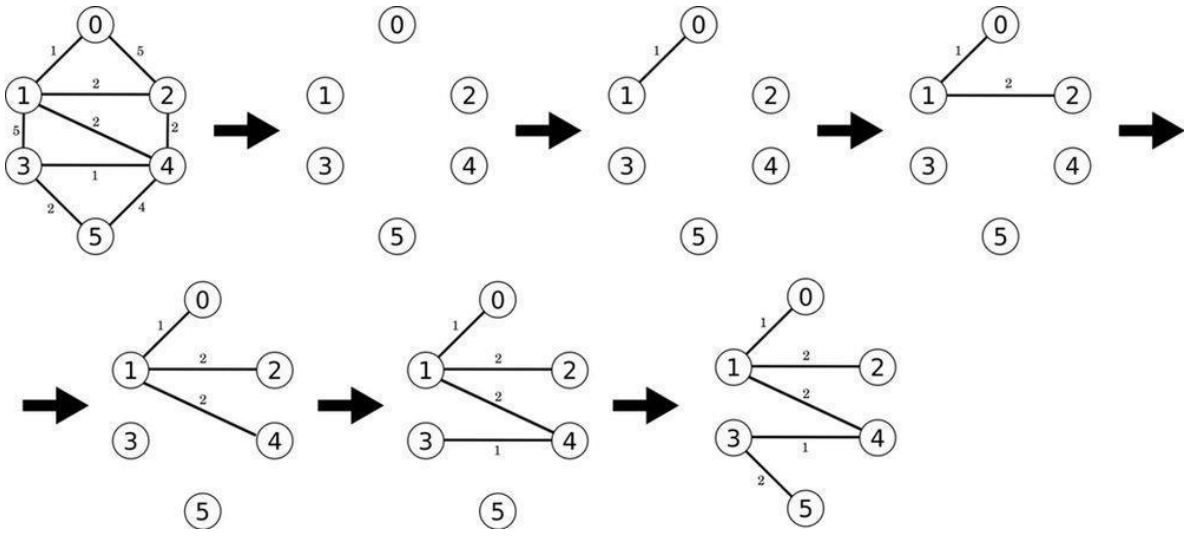


figura 2

Backtracking

Utilizamos como base el pseudocodigo explicado en la teoría (figura 3). Backtracking determina las soluciones al problema buscando sistemáticamente en el espacio de soluciones para una instancia del problema. Nuestra solución fue planteada como un arreglo de tamaño N (cantidad de estaciones), donde cada estación tenía que tener asignada un predecesor con su costo correspondiente. El algoritmo que realizamos va a tener una complejidad O(#hijos_peor_nodo^N*C)

donde "C" corresponde al costo computacional del algoritmo (es decir, lo que cuesta realizar cada iteración).

```
BACK (estado e, solucion *sol)   \\ e: nodo del árbol del espacio de soluciones
{                                \\ sol: solución que retorna
   if ( HOJA (e))
      CalcularSolución (e, sol);
   else
      {
      int nrohijo = 1;
      estado siguiente;
      while ( HIJOS (nrohijo, e, siguiente ) )
            {  if  ( !PODADO ( siguiente, sol) )
                  BACK ( siguiente, sol);
            ++nrohijo;   }
      }
}
```

figura 3.

La manera de implementar este método es utilizar como solución un arreglo de tamaño n-1 (cada vértice va a tener su vértice padre menos el origen), donde "Xi" va a contener el predecesor de i, guardando los costos en otro arreglo a parte para ir comparando.

Tabla Comparativa

Tamaño entrada	Prim	Backtracking
N=5 hijos_peor_nodo=3	5^2=25	3^5*3 = 729
N=10 hijos_peor_nodo=4	10^2=100	4^10*4 = 4194304
N=100 hijos_peor_nodo=10	100^2=1000	10^100*10=1.0E101

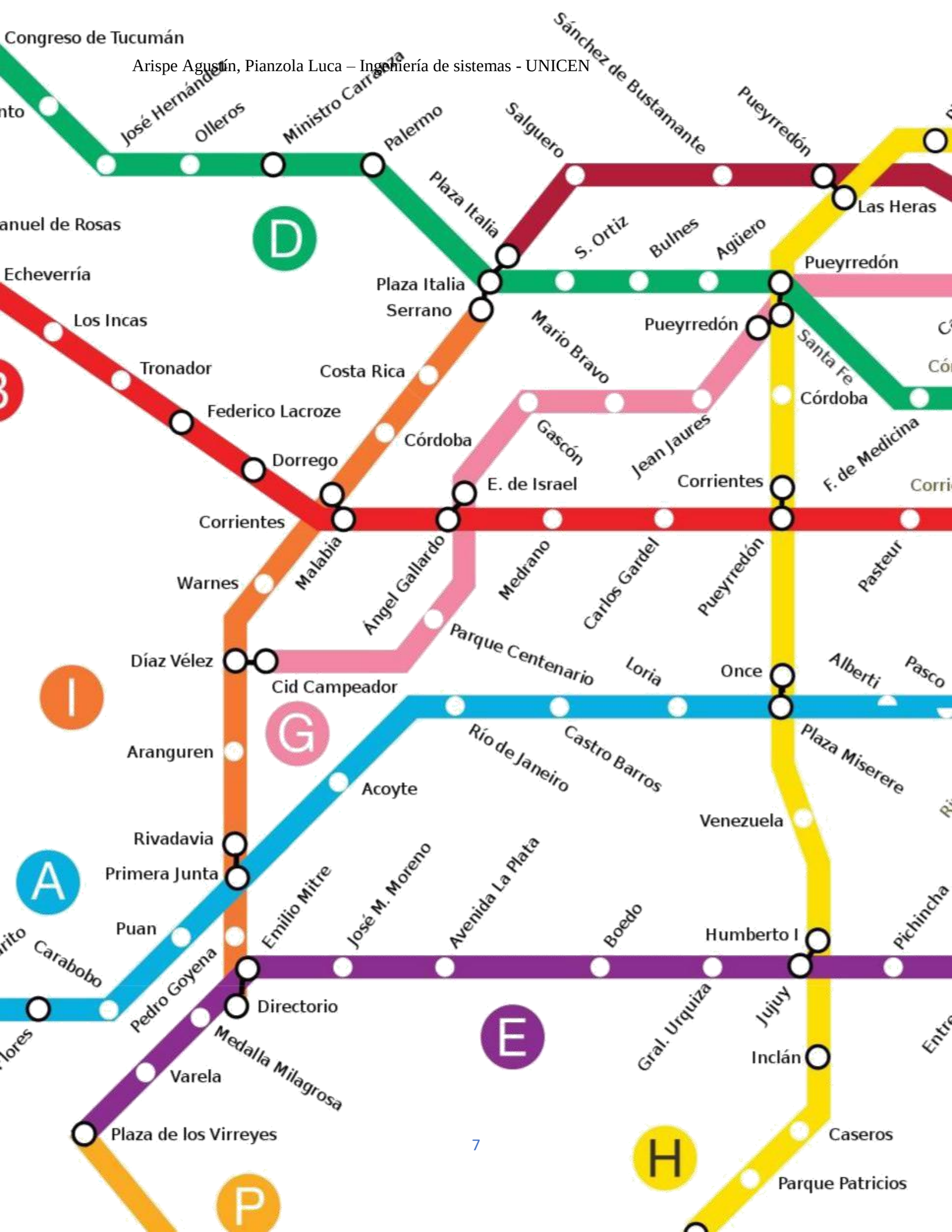
Para poder ver mejor la diferencia en los costos mostraremos una tabla con distintos tamaños de entradas.

Para el calculo real de la complejidad utilizamos “e” (cantidad de aristas del peor vértice) ya que, en estos tres casos, y en la mayoría de las posibilidades log n (cantidad de vértices) siempre va a ser menor que “e”, por lo tanto, si en las funciones de la clase grafo se realizan iteraciones de una complejidad O(max(log n, e), nosotros utilizaremos “e”.

Si miramos las complejidades reales en cada uno de los casos veremos que backtracking termina siempre siendo más factible.

Conclusión

Las dos técnicas son factibles, pero en este caso, la complejidad que calculamos en los dos casos es mejor con Backtracking.


Congreso de Tucumán
José Hernández
Olleros
Ministro Carranza
Palermo
Plaza Italia
Salguero
Sánchez de Bustamante
Pueyrredón
Las Heras
D
anuel de Rosas
Echeverría
Los Incas
Tronador
Federico Lacroze
Dorrego
Plaza Italia
Serrano
Costa Rica
Córdoba
S. Ortiz
Bulnes
Agüero
Pueyrredón
Mario Bravo
Pueyrredón
Santa Fe
Córdoba
Gascón
Jean Jaures
F. de Medicina
E. de Israel
Corrientes
Corrientes
Malabia
Ángel Gallardo
Medrano
Carlos Gardel
Pueyrredón
Pasteur
Warnes
Díaz Vélez
Cid Campeador
Parque Centenario
Loria
Once
Alberti
Pasco
I
G
Aranguren
Río de Janeiro
Castro Barros
Plaza Miserere
Acoyte
Venezuela
Rivadavia
Primera Junta
A
Puan
Emilio Mitre
José M. Moreno
Avenida La Plata
Boedo
Humberto I
Pichincha
Carabobo
Pedro Goyena
Directorio
E
Gral. Urquiza
Jujuy
Inclán
Medalla Milagrosa
Varela
Plaza de los Virreyes
H
Caseros
Parque Patricios
P