Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre: Enero-Marzo 2024

Un algoritmo Divide-and-Conquer para el problema de la ruta de máximo beneficio

Integrantes:

19-10040 → Revete, Jose
1910109 → De Vincenzo, Alejandro

Diseño de la solución:

La solución para el algoritmo se divide en distintas partes, en primera instancia se define la función main la cual le dará lugar a la ejecución del resto de funciones. Esta funcion main tendrá como primera tarea la lectura y recolección de los datos del archivo recibido en la ejecución del script de bash. Luego se observan todas las primeras y segundas coordenadas para definir el mapa sobre el cual se estará trabajando. Por consiguiente se dará lugar a la ejecucion de la funcion divideAndConquerPRMB la cual tendrá como primera labor el verificar si existen más de 3 pares ordenados en x tal que tengas sus primeras coordenadas iguales, luego divide en 3 casos:

1. Si la cantidad de ciudades está entre 1 y 3, entonces ejecuta la función resolverMiniPRMB, la cual se encargará de hacer una prueba manual se todas casos presentes dependiendo de la cantidad de ciudades ingresadas, ejemplo: si son 2 ciudades, deberá comprobar la recompensa en cada ciudad y la ganancia que trae el ir de una ciudad a otra, esto lo realiza llamando a las funciones gananciaRuta y está a la función costo (está calcula el costo que implica ir de una ciudad a otra). Luego esta función llamará a la función unirRutasCasos, la cual se encargará de combinar las coordenadas y strings de rutas obtenidas, para así registrar los datos necesarios de las ciudades y llamar a otra función. Esta nueva función se llama escogerVecino, este algoritmo prueba cada elemento de la ruta previamente obtenida al unir, define una coordenada para calcular la distancia entre esta ciudad y las demás para al final escoger la que implica menor costo, esto se realiza mediante las funciones costoRuta y distancias, luego al finalizar de recorrer los elementos de la ruta se ejecuta la función todasVisitadas, para verificar que todas la ciudades han sido visitadas. Luego como todas las rutas y ganancias se iban comparando mientras se ejecutaba el algoritmo, entonces obtener la ruta con mejor ganancia y la mejor ganancia.
2. Si la cantidad de ciudades es 0, no se hace nada.
3. Si la cantidad de ciudades es mayor que 3 entonces dividimos el plano por la mitad y aplicamos recursión para evaluar estos dos planos que hemos conseguido. Posteriormente, si ahora la cantidad de ciuadad es menor o igual que 3, entonces se llama a la función resolverMiniPRMB y se desarrolla el caso expuesto en 1., pero si la cantidad de ciudades es mayor que 3, entonces se sigue llamando recursivamente a divideAndConquerPRMB.

Algoritmo 1: Divide-and-Conquer
Entrada: Una instancia x de un problema.
Salida : Una solución y de la instancia x

```
1       Si ciudades es suficientemente pequeño entonces
2               devolver adhoc;
3
4       Si ciudades es 0 entonces
5               // No hay ciudades, no se hace nada
6
7       Sino
8               Si hay más de 3 ciudades en la misma coordenada en el eje "x" entonces
9                       divisionEnY = Verdadero;
10              Sino
11                      divisionEnY = Falso;
12
13      Si divisionEnY es Verdadero entonces
```

```
14                  Se divide el plano en 2 partes iguales en el eje Y;
15                  Se guardan las coordenadas de las ciudades que tienen coordenadas[1] <=
maxYmitad; // maxYmitad es el punto en donde hizo la division del plano
16                  Se guardan coordenadas de las ciudades que tienen coordenadas[1] > maxYmitad;
17                  Se guarda la recompensa correspondiente a las ciudades tal que coordenadas[1] >
maxYmitad;
18                 Se guarda la recompensa correspondiente a las ciudades tal que coordenadas[1] <=
maxYmitad;
19                  rutaArriba = Divide-and-Conquer; // Se aplica la recursion para las divisiones del
plano;
20                  rutaAbajo = Divide-and-Conquer; // Se aplica la recursion para las divisiones del
plano;
21                  // Algoritmo de unirRutas
22
23        Sino
24                  Se divide el plano en 2 partes iguales en el eje X;
25                  Se guardan las coordenadas de las ciudades que tienen coordenadas[0] <=
maxXmitad; // maxYmitad es el punto en donde hizo la division del plano
26                  Se guardan las coordenadas de las ciudades que tienen coordenadas[0] > maxXmitad;
27                  Se guarda la recompensa correspondiente a las ciudades tal que coordenadas[0] <=
maxXmitad;
28                  Se guarda la recompensa correspondiente a las ciudades tal que coordenadas[0] >
maxXmitad;
29                  rutaIzq = Divide-and-Conquer; // Se aplica la recursion para las divisiones del plano;
30                  rutaDer = Divide-and-Conquer; // Se aplica la recursion para las divisiones del plano;
31                  // Algoritmo de unirRutas
32
33    devolver ruta;
```

Algoritmo 2 : adhoc
Entrada: Una instancia x de Divide-and-Conquer.
Salida : Una solución y de la instancia x tal que se obtiene la ruta la ganancia de x.

```
1         Si el número de ciudades es 1 entonces
2               Calcular la ganancia para una sola ciudad:
3               Se genera y, que es la ruta y la ganancia:
4         Si el número de ciudades es 2 entonces
5                   Calcular la ganancia para dos ciudades;
6                   Si la ganancia de la primera ciudad es mayor que la segunda entonces
7                               Se genera y, la ruta y la ganancia de la primera ciudad;
8                   Sino
9                               Se genera y, la ruta y la ganancia de la segunda ciudad;
10        Si el número de ciudades es 3 entonces
11                  Calcular la ganancia para todas las combinaciones de tres ciudades;
12                  Elegir la combinación con la mayor ganancia;
13                  Se genera y, la ruta y la ganancia de esa combinación;
14        devolver y;
```

Detalles de la implementación:

La realización del algoritmo fue una prueba y corrección constante. Luego de pensar en varias formas de llevar a cabo el proyecto, la mejor implementación del algoritmo que conseguimos es la que se puede apreciar en el Proyecto. Este funciona bien hasta cierto límite, se probó con 400 ciudades y tarda un aproximado de 4 min el concluir la prueba. En pocas ciudades el algoritmo funciona bien y el tiempo de respuesta es de menos de 1 min.

Lecciones aprendidas:

Durante la realización del proyecto la mayor lección aprendida fue la de descansar y no presionar de más para completar algo, esto en varias ocasiones resultó contraproducente en cuanto a la intención del algoritmo. Además, investigar, intercambiar de ideas y opiniones fue vital, dado que sin el apoyo mutuo, muy probablemente no pudiéramos haber concluido este proyecto. Por último, no tener miedo a probar mil metodos distintos nos ayudó a avanzar o nos enseñó errores que luego no llegamos a cometer.