Esame laboratorio di algoritmi

Docente: Marinella Sciortino

Francesco La Rosa **0696471**

Dario Zappata **0699471**

Obiettivo

Il problema in analisi riguarda la comunicazione tra avamposti che può avvenire mediante canali satellitari e/o canali radio. Due avamposti con canali satellitari possono comunicare direttamente tra loro, mentre due avamposti con canali radio per potere comunicare devono essere posti a una distanza D che dipende dalla potenza del ricetrasmettitore radio. Dati in input il numero di canali satellitari, il numero di avamposti e le loro coordinate vogliamo stabilire come assegnare i canali satellitari ai rispettivi avamposti in maniera tale da diminuire il più possibile il valore di D, e di conseguenza risparmiare sull'acquisto dei trasmettitori radio.

Strategia risolutiva

Per risolvere tale problema abbiamo acquisito l'input sotto forma di grafo: gli avamposti vengono rappresentati come nodi e la loro distanza raffigura il peso degli archi che congiunge gli avamposti. Per determinare la distanza minima che permette di collegare gli avamposti, abbiamo deciso di calcolare il minimo albero ricoprente del grafo. Una volta computato il MST, al fine di calcolare un valore di D minimo per collegare gli avamposti, assegniamo i canali satellitari agli estremi degli archi di peso maggiore.

Acquisizione dell'input

La funzione main consente di acquisire più test case specificando per ognuno di essi il numero di satelliti, di avamposti e le coordinate di essi; verificando che ogni inserimento rispetti le indicazioni del problema. Nel caso di input corretto viene creato un grafo completo con numero di nodi pari al numero di avamposti e con ogni vertice collegato direttamente a tutti gli altri vertici per un totale di n*(n-1)/2 archi. Ciascun arco sarà etichettato con la distanza euclidea tra le coordinate dei vari avamposti. Tale grafo è implementato mediante una lista delle adiacenze.

Operazione	Complessità spaziale	Complessità temporale
Creazione grafo	O(V+E)	O(V+E)

Calcolo del MST

Per determinare il modo ottimale che consente di collegare gli avamposti, abbiamo deciso di calcolare il MST ottenendo l'albero che connette gli avamposti con distanza minima. Tra i vari algoritmi che consentono di calcolare il MST abbiamo deciso di utilizzare eager Prim che risulta essere più efficiente là dove il numero dei vertici è minore del numero degli archi (come nel caso in esame). L'approccio greedy per il calcolo del MST risulta essere corretto e ciò è garantito dal seguente teorema: Dato un qualsiasi taglio in un grafo, il crossing edge di peso minimo sta nel MST. Diversi algoritmi per il calcolo del MST si caratterizzano per la scelta del cut e per l'estrazione del crossing edge di peso minimo, nello specifico Prim ad ogni passo seleziona come cut l'insieme dei vertici appartenenti al MST attualmente computato e l'insieme contenente i restanti vertici; tra i vari crossing edge che hanno un solo estremo in T sceglie quello di peso minimo.

Nello specifico per implementare eager Prim abbiamo utilizzato le seguenti strutture dati:

- Un vettore di interi **edgeTo** di dimensione V per tenere traccia del vertice di partenza dell'arco che inseriamo nel MST. Ad esempio edgeTo[v]=w indica che giungiamo a v nel MST tramite un arco che parte da w.
- Un vettore di double **distTo** di dimensione V che indica il peso dell'arco indicato da edgeTo.
- Un vettore di bool **marked** di dimensione V che tiene traccia dei vertici appartenenti al MST, al fine di evitare un arco che creerebbe un ciclo.
- Una coda a priorità **pq** sui vertici di dimensione V per effettuare l'estrazione del minimo e per aggiornare eventualmente il costo dell'arco che porta a quel vertice.
- Una coda a priorità **maxEdge** sui vertici di dimensione V che tiene traccia degli archi appartenenti al MST ordinati per chiave massima, necessario per l'assegnazione dei canali satellitari.

Per le due code a priorità usiamo la classe **PriorityQueue** da noi creata, il costruttore di tale classe oltre alla dimensione, prende a parametro un valore booleano che ci permette di stabilire se gestire l'heap per massimo o per minimo. L'oggetto di tipo PriorityQueue contiene 3 array interni: l'array **pq** rappresenta il nostro heap in cui il primo elemento (il max o il min) si trova nell'indice 1, l'array **qp** dove qp[i] indica in che posizione si trova il vertice i nell'heap, l'array **key** in cui key[i] indica il costo per arrivare al vertice i. (N.B: per gli array qp e key il primo valore effettivo si trova nell'indice 0).

La rimozione dell'elemento i dalla coda a priorità comporta l'inserimento della macro INF in qp[i], in questo modo comprendere se un elemento non è presente nella coda a priorità costa O(1) piuttosto che effettuare la ricerca nell'heap che nel caso pessimo richiede O(n).

Operazione	**Complessità spaziale**	**Complessità temporale**	**Spiegazione**
Costruzione delle due PQ	O(V)	O(VlogV)	V operazioni di inserimento di costo logV
Estrazione del minimo	------	O(VlogV)	V operazioni di cancellazione di costo logV
Cambio di priorità	------	O(ElogV)	Nel caso pessimo effettuiamo E cambiamenti di priorità di costo logV

Le operazioni di inserimento, cancellazione e cambio di priorità nella coda a priorità per vertici implementata con heap costano O(logV). L'operazione di ricerca (contains nel codice) nella coda a priorità avviene in tempo O(1).

In totale nel caso pessimo l'operazione del calcolo del MST richiede **O(ElogV)**.

Assegnazione dei canali satellitari

La scelta dei nodi a cui assegnare i canali satellitari in modo tale da diminuire il valore minimo di "D" viene effettuata mediante l'estrazione dell'arco di peso massimo dal MST, i cui estremi verranno scelti per l'istallazione del canale satellitare. L'estrazione dell'arco di peso massimo avviene per mezzo dei vettori **edgeTo** e **maxEdge** i quali valori sono stati impostati durante l'esecuzione dell'algoritmo di Prim.

```
int w=this->maxEdge.getRoot();
int v=this->edgeTo[w];
```

Grazie alle istruzioni precedenti riusciamo a determinare l'arco v->w di peso massimo del MST. Una volta ottenuto tale arco, per mezzo del vettore **satellite** di valori booleani, assegniamo i canali satellitari ai vertici estremi. Iteriamo questo processo fintantoché non esauriamo il numero di canali satellitari da distribuire.

Correttezza:

Una volta calcolato il MST, nel caso in cui non dovessimo attribuire alcun canale satellitare, la scelta della distanza minima che tutti gli avamposti devono ricoprire per la comunicazione radio coincide con la distanza maggiore che collega due avamposti. Questa tesi suggerisce di assegnare i canali satellitari agli avamposti più distanti, in questo modo attribuiti 2 canali satellitari ai vertici dell'arco di peso massimo la distanza minima coinciderà con il successivo arco di peso massimo del MST ovvero con la successiva distanza massima che la comunicazione radio deve ricoprire.

Proviamo allora che la distanza minima coincida con l'arco di peso maggiore del MST, ricordando che il MST connette tutti i vertici del grafo con costo minimo dove il costo nel nostro caso rappresenta la distanza tra gli avamposti. Assumiamo per assurdo che la D minima sia data dal costo di un arco "**e**" che non sia quello di costo massimo "**f**" nel MST. Ciò significa che a tutti gli avamposti verrà assegnato lo stesso trasmettitore radio con potenza in grado di ricoprire la distanza data dal costo dell'arco "e", essendo il costo di "e" minore del costo di "f" gli avamposti corrispondenti ai vertici estremi di "f" con un tale trasmettitore non riuscirebbero a comunicare. Viceversa con un trasmettitore in grado di ricoprire la distanza massima del MST tutti gli avamposti saranno in grado di comunicare.

Dato in input S, il numero di canali satellitari da assegnare, denotiamo il costo dell'algoritmo:

Operazione	Complessità spaziale	Complessità temporale	Spiegazione
Estrazione del massimo	------	O(SlogV)	Nel caso pessimo, ovvero quando estraiamo ogni volta dalla coda un arco di cui un estremo è marcato, l'operazione di estrazione viene effettuata esattamente S-1 volte ciascuna di costo logV.
Marcatura satelliti	O(V)	O(S)	Usiamo un array di dimensione V, la verifica se un arco è marcato da un satellite avviene al più esattamente S volte.

In totale nel caso pessimo l'operazione del calcolo della distanza minima richiede **O(SlogV)**.

Nel complesso al fine di risolvere il problema in esame, calcoliamo il MST in O(ElogV) e la distanza minima in O(SlogV) il chè significa che prevale la complessità temporale dell'algoritmo di Prim **O(ElogV)**.

Test su input

Abbiamo effettuato veri test per verificare il funzionamento dell'algoritmo. Di seguito riportiamo 5 test dati in input al nostro programma (nell'output di seguito sono presenti stampe ulteriori per verificare la corretta creazione del grafo e calcolo del MST, lanciando il programma sarà solo visualizzata la D minima per ogni test).

- Test 1:

Input	Output
2 4 0 100 0 300 0 600 150 750	Lista delle adiacenze 0: 3 -> 667.08 2 -> 500.00 1 -> 200.00 1: 3 -> 474.34 2 -> 300.00 2: 3 -> 212.13 3: MST 0 -> 1 c: 200.00 1 -> 2 c: 300.00 2 -> 3 c: 212.13 D minima: 212.13

- Test 2:

Input	Output
4 5 66 10 900 310 36 60 150 750 65 250	Lista delle adiacenze 0: 4 -> 240.00 3 -> 744.75 2 -> 58.31 1 -> 886.32 1: 4 -> 837.15 3 -> 869.54 2 -> 899.44 2: 4 -> 192.20 3 -> 699.35 3: 4 -> 507.17 4: MST 0 -> 2 c: 58.31 2 -> 4 c: 192.20 2 -> 3 c: 699.35 0 -> 1 c: 886.32 D minima: 192.20

- Test 3:

Input	Output
4 4 237 12 356 15 67 111 12 35	Errore nel test 3! Inserire un numero di avamposti maggiore del numero di satelliti e minore di 500!

- Test 4:

Input	Output
3 5 22 34 66 35 112 47 175 134 0 120	Lista delle adiacenze 0: 4 -> 88.77 3 -> 182.78 2 -> 90.93 1 -> 44.01 1: 4 -> 107.62 3 -> 147.25 2 -> 47.54 2: 4 -> 133.69 3 -> 107.42 3: 4 -> 175.56 4: MST 0 -> 1 c: 44.01 1 -> 2 c: 47.54 0 -> 4 c: 88.77 2 -> 3 c: 107.42 D minima: 88.77

- Test 5:

Input	Output
5 8 20 30 45 130 27 840 521 332 115 900 750 23 661 427 500 500	Lista delle adiacenze 0: 7 -> 671.79 6 -> 753.98 5 -> 730.03 4 -> 875.17 3 -> 584.98 2 -> 810.03 1 -> 103.08 1: 7 -> 586.45 6 -> 683.86 5 -> 713.07 4 -> 773.18 3 -> 517.09 2 -> 710.23 2: 7 -> 582.52 6 -> 756.65 5 -> 1090.97 4 -> 106.51 3 -> 708.59 3: 7 -> 169.31 6 -> 169.19 5 -> 384.61 4 -> 698.18 4: 7 -> 555.18 6 -> 722.39 5 -> 1082.75 5: 7 -> 538.54 6 -> 413.69 6: 7 -> 176.78 7: MST 0 -> 1 c: 103.08 1 -> 3 c: 517.09 3 -> 6 c: 169.19 3 -> 7 c: 169.31 3 -> 5 c: 384.61 3 -> 4 c: 698.18 1 -> 2 c: 710.23 D minima: 169.31