Relazione Progetto 2

Ingegneria degli Algoritmi 2018/2019

Matteo Lungo – 0253524

**Scelte Implementative**

Per la realizzazione del progetto, la struttura Grafo è stata modificata implementando i metodi **dHeapPrioritySearch**e **binomialHeapPrioritySearch** all’interno della classe **GraphBase** e il parametro **weight** all’interno della classe **Node** (file Graph.py) secondo le istruzioni fornite dal testo. All’ interno dei due metodi di visita, il set di nodi da esplorare è stato rimpiazzato rispettivamente con un D-Heap (PQ\_DHeap.py) e un Heap Binomiale (PQbinomialHeap.py). I nodi visitati vengono infine mostrati tramite il print della lista **markedNodes**. Per l’inizializzazione e l’estrazione del valore massimo (corrispondente al peso del nodo), le due strutture sono state adeguatamente modificate. In particolare, i metodi modificati sono:

Per D-Heap:

* **minSon → maxSon**
* **findMin → findMax**
* **deleteMin → deleteMax**
* **insert**

Per Heap Binomiale:

* **findMinIndex → findMaxIndex**
* **findMin → findMax**
* **deleteMin → deleteMax**
* **insert**
* aggiunto metodo **lenght**

Per la generazione del grafo è stato ricostruito il metodo **buildGraph** del **GraphHelper** (file graphHelper.py). Il metodo in questione genera un grafo dal numero di nodi scelto a cui vengono attribuiti valori casuali. L’algoritmo genera poi il numero di archi scelto (dal valore casuale), assicurandosi che il grafo sia connesso.

**Risultati Sperimentali e Commenti**

Per l’analisi dei tempi di esecuzione degli algoritmi di visita è stata utilizzata la funzione PQSearchPerformance (file PQSearchPerformance.py). Il confronto è stato effettuato prima all’aumentare di nodi e archi in numero uguale (Grafico 1) e poi all’aumentare dei soli archi con numero di nodi costante (Grafico 2).

Nel primo caso, i tempi di esecuzione della visita aumentano esponenzialmente all’aumentare del numero di nodi/archi per ognuna delle code con priorità senza presentare troppe differenze. Un’ipotetica classifica (in ordine crescente di tempi) sarebbe:

1. 5-Heap
2. Heap Binario
3. Heap Binomiale

Nel secondo caso invece, i tempi di esecuzione della visita aumentano esponenzialmente all’aumentare del numero di archi (con numero di nodi costante), presentando valori più elevati a parità di archi rispetto al primo caso:

ES.

* 1000/1000: ~0,5 s
* 100/1000: ~16 s

I valori riportati presentano differenze trascurabili tra le varie code di priorità con i dati utilizzati.