2η Εργασία στο μάθημα Αλγόριθμοι &Δομές Δεδομένων

Ον/μο : Μάριος-Χρήστος Σταματίου

A.M : 1066488

Έτος: 3ο

Πρόβλημα προς επίλυση: Στο πρόβλημα που μας δόθηκε έχουμε να παράξουμε 5 κώδικες , οι οποίοι θα δημιουργούν 800 κόμβους μέσω των οποίων θα πραγματοποιούνται 10.000 ταξίδια , όπου το κάθε ταξίδι θα έχει και διαφορετικό βάρος- των οποίων το πεδίο τιμών θα είναι από 50 μέχρι 100. Το όριο υλοποίησης ενός ταξιδιού θα είναι 200.

ΣΗΜΕΙΩΣΗ: Σε κάθε ένα από τους 6 κώδικες έχω συμπεριλάβει μια main στην οποία αρχικοποιώ τα δεδομένα που χρειάζονται, τον χρόνο έναρξης και λήξης υπολογισμού των πράξεων καθώς και τις απαραίτητες επιπλέον μεθόδους που ζητούνται ανά ερώτημα, ώστε το πρόγραμμα να είναι οργανωμένο και εύκολο ως προς την ανάγνωση/κατανόηση.Επίσης,για του κόμβους και τα ταξίδια έχω βάλει input() , ώστε ο διορθωτής να μπορεί να ελέγξει ευκολότερα την ορθότητα των αποτελεσμάτων (σε περίπτωση που χρειασθεί να εισάγει λιγότερους κόμβους και ταξίδια). Η επίλυση των προβλημάτων έχει υλοποιηθεί με τη γλώσσα Python(v.3.9.1)

Α) Ξεκινώντας , από τον κώδικα του ερωτήματος Α παράγουμε 2 συναρτήσεις (SparseGenerator & CreateArrays) για να διαιρέσουμε το πρόβλημα μας σε υπο-προβλήματα (πιο απλό και ευκολότερο στην διόρθωση/διάβασμα). Στη συνάρτηση SparseGenerator δημιουργούμε 4 λίστες οι οποίες είναι για :

1)Αρχικοποίηση κόμβων

2)Έλεγχος κόμβων που έχουμε επισκευθεί

3)Εισαγωγή των βαρών μεταξύ 2 κόμβων

Έπειτα , επιλέγουμε τυχαίο δείγμα (curVertex) , αφαιρώντας το από το InitialSet ενώ παράλληλα το τοποθετούμε στο VisitedSet και καλούμε την CreateArrays() στην οποία δημιουργούνται οι πίνακες περιγραφής της Αραιής Μήτρας (jloc,jval,irow).

Στην CreateArrays , λοιπόν , με μια while-loop παίρνουμε τυχαία κόμβους από το InitialSet και παράγουμε τα τυχαία βάρη , και έπειτα με ένα if-else statement θέτουμε του περιορισμούς μας για το ολοκληρωμένο ταξίδι (limit=200).

Β) Στο ερώτημα Β ακολουθούμε επακριβώς την λογική του ερωτήματος Α , με τη μόνη διαφορά ότι προσθέτουμε έναν πίνακα Next που μετράει τα ταξίδια και στο τέλος του κάθε ενός τοποθετούμε την τιμή -1 για να υποδείξουμε τη λήξη του (κάθε ταξιδιού).

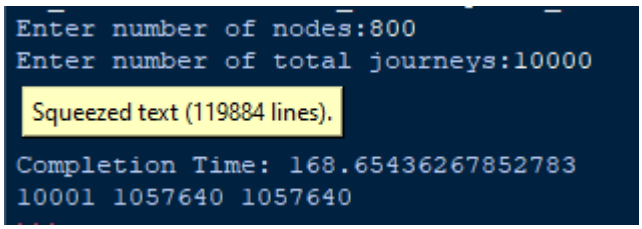
1)Στο ερώτημα αυτό , προσθέτουμε στην CreateArrays ένα if-statement εντός του while loop και ένα εκτός στο τέλος της συνάρτησης και ανάλογα με την είσοδο Start/Finish επιλέγουμε πότε θα προσθέσουμε το ταξίδι στις λίστες μας.

Στην replacementArrayElements επαναλαμβάνουμε την τακτική που είδαμε στα ερωτήματα Α,Β και με οδηγό την λίστα Next ελέγχουμε πότε θα κάνουμε αντικατάσταση στους πίνακες (από την Next παίρνουμε το index για το που θέλουμε να τοποθετήσουμε το ταξίδι).Η replace είναι απλώς μια συμπύκνωση των μεθόδων .pop(), .insert() για μία λίστα.

2)Στο συγκεκριμένο ερώτημα αξιοποιούμε τον κώδικα του ερωτήματος Α και δημιουργούμε την συνάρτηση create_ATA_array() η οποία παίρνει ως ορίσματα τους πίνακες a_jloc & a_irow . Αρχικοποιούμε τον counter (c) , καθώς και τους πίνακες που μας ζητούντε (b_iloc,b_jloc,b_jval) . Έπειτα , με μια for-loop υπολογίζουμε τον πίνακα ΑΤ και παράλληλα με μία δεύτερη for-loop σκανάρουμε τον πίνακα ΑΤ και προσθέτουμε κάθε φορά τα στοιχεία στους κατάλληλους πίνακες . Έτσι , επιστρέφουμε αυτούς στο τέλος της συνάρτησης.

Όπως παρατηρούμε , η συνάρτηση υπολογισμού της Μήτρας Β είναι πολυπλοκότητας O(n^2).

Τέλος, τα μεγέθη των πινάκων b_iloc , b_jloc , b_jval αναφέρονται στην παρακάτω εικόνα με την αντίστοιχη σειρά .



3)Στο 3ο ερώτημα εισάγουμε την βιβλιοθήκη collections με την οποία μέσω της συνάρτησης leastVisitedNode επιστρέφουμε ένα λεξικό , το οποίο κάνουμε sort στην main και έτσι επιλέγουμε πρώτο key-value που αποτελεί τον κόμβο με τις λιγότερες επισκέψεις και τον αριθμό αυτών.

4)

ΕΡΩΤΗΜΑ 2^{Ο} :

```
= RESTART: C:\Users\stama\OneDrive\Desktop\Ηλ.Μηχανικών\3ο Έτος ΗΜΤΥ\6ο Εξάμηνο\Αλγόριθμοι_και_Δ
εις_&_Παραδοτέα\Παραδοτέο(2)_Algorithms_&_Data_Structures\Source_Codes\Sparse_Technology_(2).py
Enter number of nodes:800
Enter number of total journeys:10000
Squeezed text (82681 lines).
Completion Time: 147.52617597579956

>>>
= RESTART: C:\Users\stama\OneDrive\Desktop\Ηλ.Μηχανικών\3ο Έτος ΗΜΤΥ\6ο Εξάμηνο\Αλγόριθμοι_και_Δ
εις_&_Παραδοτέα\Παραδοτέο(2)_Algorithms_&_Data_Structures\Source_Codes\Sparse_Technology_(2).py
Enter number of nodes:800
Enter number of total journeys:20000
Squeezed text (369422 lines).
Completion Time: 726.382269859314
```

Παρατηρούμε , πως με την εισαγωγή διπλάσιων ταξιδιών ο χρόνος ολοκλήρωσης των πράξεων σχεδόν τετραπλασιάζεται (κάτι που είναι απολύτως λογικό και επαληθεύει την πολυπλοκότητα , 4*147.5=590 ~726)

ΕΡΩΤΗΜΑ 3^{Ο} :

```
= RESTART: C:\Users\stama\OneDrive\Desktop\Παραδοτέο(2)_Algorithms_&_Data_S
(3).py
Enter number of nodes:800
Enter number of total journeys:10000
Squeezed text (3479 lines).
Least Visited Node:287 , Times Visited:1
Completion Time: 4.680931091308594
>>>
```

```
= RESTART: C:\Users\stama\OneDrive\Desktop\Παραδοτέο(2)_Algorithms_&
(3).py
Enter number of nodes:800
Enter number of total journeys:20000
Squeezed text (6986 lines).
Least Visited Node:287 , Times Visited:1
Completion Time: 11.330377340316772
>>>
```

Παρατηρούμε πως ο χρόνος σχεδόν τριπλασιάζεται με τον διπλασιασμό των συνολικών ταξιδιών για το ερώτημα 3.