

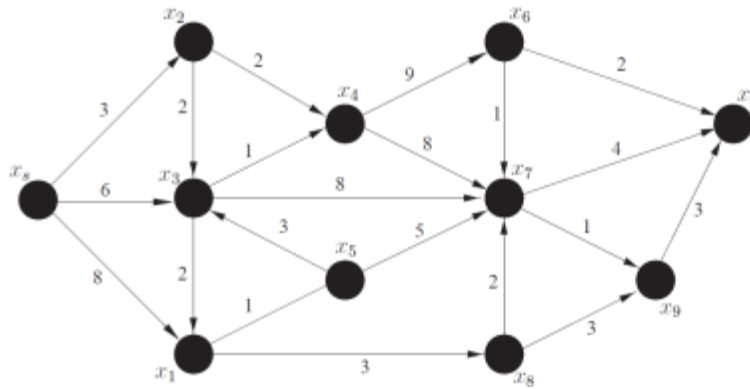
Συνδυαστική Βελτιστοποίηση

Εργαστηριακή αναφορά

Άσκηση 1

Σοφιανόπουλος Έκτορας 2017010016

Να γραφεί πρόγραμμα σε Matlab, Octave, C, C++, Python, Java... που να εφαρμόζει τον αλγόριθμο Dinic ή τον αλγόριθμο Ford-Fulkerson για την επίλυση του προβλήματος της μέγιστης ροής σε οποιοδήποτε γράφημα (για παράδειγμα στο ακόλουθο γράφημα). Στην αναφορά να περιγράφετε τον αλγόριθμο που αναπτύξατε αναλυτικά και να δίνονται, όπως και στο πρόγραμμα, η συνολική ροή και οι διαδρομές που θα ακολουθηθούν.




Αρχικά κατασκευάστηκε η **συνάρτηση read_graph** η οποία δέχεται τις ακμές του γραφήματος ως “αρχικό node τελικό node capacity” και τις αποθηκεύει σε ένα dictionary. Η εντολή defaultdict από τη βιβλιοθήκη collections χρησιμεύει καθώς μπορούμε να προχωρήσουμε στην τροποποίηση της χωρητικότητας μιας ακμής χωρίς να χρειαστεί να ελέγξουμε πρώτα αν είναι καταχωρημένη στο dictionary. Η συνάρτηση διαβάζει κάθε γραμμή του input.txt και αποθηκεύει τα στοιχεία της ως : `graph[start][end]=int(capacity)` δημιουργώντας έτσι το γράφημα. Επιπλέον η συνάρτηση επιστρέφει όλα τα nodes (`all_nodes`), συμπεριλαμβανομένου και του τελευταίου από το οποίο δεν ξεκινούν ακμές.

Η συνάρτηση `ford_fulkerson` δέχεται ως ορίσματα το `graph`, την μεταβλητή `nodes` που βοηθά στην καταγραφή των κόμβων που έχουμε ήδη επισκεφτεί, καθώς και τον αρχικό και τον τελικό κόμβο, οι οποίοι για την διευκόλυνση του χρήστη θα είναι πάντα οι `start : "S"` και `sink : "T"`. Χρησιμοποιεί BFS (αναζήτηση κατά πλάτος) για την εύρεση διαδρομών ως επαναληπτική διαδικασία που σταματά όταν δεν υπάρχουν νέες διαδρομές. Οι κόμβοι που πρέπει να επισκεφτούμε είναι στην ουρά `"queue"` (χρησιμοποιεί την `deque` δομή δεδομένων η οποία επιτρέπει την αφαίρεση και προσθήκη στοιχείων και από τα δύο άκρα της). Η εντολή `queue.popleft()` αφαιρεί και επιστρέφει τον πρώτο κόμβο. Κάθε επανάληψη `"for v, capacity in graph[u].items()"` διατρέχει όλους τους γειτονικούς (`v`) κόμβους του `u`, αφού ελέγξει ότι η διαφορά του `capacity` με την ροή είναι θετική. Όταν βρεθεί διαδρομή προς το `sink node` η συνάρτηση επιστρέφει `True` και ενημερώνει την ροή και καταγράφει τα `nodes-parents` κάθε κόμβου στο `dictionary parent` ώστε να ανακατασκευάσει τη διαδρομή αργότερα. Η συνάρτηση `ford_fulkerson` επιστρέφει τη μέγιστη ροή που μπορεί να διαχειριστεί το σύστημα / γράφημα η οποία αρχικοποιήθηκε ίση με το 0 και ενημερωνόταν σε κάθε επανάληψη όπου βρισκόταν διαδρομή προς το `sink`. Επιστρέφει επίσης όλες τις επιτυχημένες διαδρομές, από το `"S"` στο `"T"`.

Η `if __name__ == "__main__"` εκτελεί τον κώδικα εφόσον το script τρέχει ως κύριο πρόγραμμα. Πρώτα διαβάζει το γράφημα από το `input1.txt` και ύστερα καλεί την συνάρτηση. Τελικά ανοίγει το αρχείο εξόδου `output1.txt` και τυπώνει μέσα ότι επέστρεψε η συνάρτηση δηλαδή την μέγιστη ροή καθώς και τις διαδρομές που ακολουθήθηκαν. Για το τελευταίο χρησιμοποιήθηκε ένας βρόγχος `for` όπου τυπώνει όλους τους κόμβους για κάθε `"path"` αποθηκευμένο στα `"paths"` διαχωρισμένους με βελάκια σε μια συμβολοσειρά.

Το script τρέχει για οποιοδήποτε γράφημα επιθυμεί ο χρήστης, αρκεί να τροποποιήσει τα περιεχόμενα του αρχείου εισόδου.

 input1 - Σημειωματάριο

Αρχείο	Επεξεργασία	Μορφή	Προβολή	Βοήθεια
Α	A	3		
S	B	6		
S	C	8		
A	B	2		
A	D	2		
B	D	1		
B	G	8		
C	E	1		
C	H	3		
D	F	9		
D	G	8		
E	B	3		
E	G	5		
F	G	1		
F	T	2		
G	T	4		
G	I	1		
H	G	2		
H	I	3		
I	T	3		

Περιεχόμενα αρχείου εισόδου.