

Diskr. i komb. met. za rač. gr. 1→15, 2→5, 3→10, 4→10, 5→5, 6→15, 7→10, 8→10.

1. Napisati algoritam za sortiranje biranjem, takozvani SELECTION SORT.

Za algoritam SELECTION SORT iz zadatka 1, za niz dužine n , neka je $S(n)$ broj zamena i $P(n)$ broj poređenja.

2. Za niz $[6, 1, 2, 3, 4, 5]$ naći $S(n)$ i $P(n)$.
3. Koliko je $S(n)$ i $P(n)$ za obrnuto sortirani ulazni niz dužine n algoritma SELECTION SORT?

4. Dati definiciju "malog o " ponašanja i pokazati da je $\frac{1}{4}n \ln n + 40n - 10 = o(n^2)$.

Za niz dužine n neka je $T_{WM}(n)$ najgori slučaj vremena sortiranja Merge sort algoritmom i $T_{WS}(n)$ najgori slučaj vremena sortiranja Selection sort algoritmom. Da li je $T_{WM}(n) = o(T_{WS})$?

Da li je $\frac{3}{4}n^2 + 3n\sqrt{n} = o(n^2 \ln n)$?

5. Nacrtati usmereni graf G koji je dat tabelom listi susedstva:

u	Adj(u)				
0	1, 4	⊙	⊙	⊙	⊙
1	2, 6				
2	3, 7				
3	7				
4	1				
5	4	⊙	⊙	⊙	⊙
6	2				
7	3, 6				

6. Na graf G primeniti DFS algoritam, kod čvorova napisati d i f vrednosti, kod grana napisati tip (TBCF), napraviti tabelu zagrada. Ako je dati graf DAG, dati topološko sortiranje čvorova, ako nije, dati graf komponenti jake povezanosti.

7. Napisati kod funkcije enqueue_list koja unosi čvor na kraj liste susedstva grafa. Napisati deo koda za unos grafa G (iz zad. 5) u okviru procedure main u niz listi susedstva grafa G leksikografski.

8. U tabeli su date cene prevoza između 5 gradova.

(a) Polazeći od čvora 1, metodom najjeftinijeg suseda naći približno rešenje problema trg. putnika (TSP).

(b) Za isti problem naći Mađarskom metodom angažovanje koje je rešenje relaksi-

ranog TSP. Komentarisati rešenja (a) i (b).

	1	2	3	4	5
1	-	7	12	13	8
2	8	-	4	9	9
3	14	5	-	10	6
4	12	8	12	-	7
5	5	9	7	6	-

```
#define max_cv 50
typedef struct _node gnode;
typedef gnode *grana;
struct _node
{
    int data;
    gnode *next;
};

void enqueue_list(grana **gp, cvor d)
{
    // Ovaj kod napisati
}

int main(void)
{
    grana G[max_cv], GT[max_cv];
    int i, n; grana *rear[max_cv];

    for (i=0; i<max_cv; i++){
        G[i] = NULL;
        rear[i] = &(G[i]);
    }
    enqueue_list(&rear[0], 1);
    // nastaviti dalje,
    // uneti ostatak grafa
    return 0;
}
```