

Operacije sa skupovima

- Napisati funkciju koja će pronaći i ispisati presjek dvaju skupova cijelih brojeva, te ustanoviti složenost algoritma.
- Razmotrit ćemo dva slučaja: kada su elementi skupova nesortirani i kada su sortirani
- Za nesortirane elemente potrebno je usporediti svaki element iz prvog skupa sa svakim iz drugog - funkcija presjek1()
- Složenost funkcije je $O(N_A * N_B)$
- Za sortirane elemente: uspoređuje se element iz prvog skupa s elementom iz drugog i pomiče se kursor kod onog skupa čiji je element manji – funkcija presjek2()
- Složenost funkcije $O(\min(N_A, N_B))$, ali radi samo za sortirane nizove

```

#include <stdio.h>
#include <stdlib.h>
#define NA 10
#define NB 15

int presjek1(int *a, int *b, int **p) {
    int i, j, k = 0; // k – broj zajedničkih elemenata
    for (i = 0; i < NA; i++)
        for (j = 0; j < NB; j++)
            if (a[i] == b[j]) {
                *p = realloc(*p, (k + 1) * sizeof(int));
                (*p)[k] = a[i]; // paziti: *p[k] znači *(p[k])!!
                ++ k; }

    return k; }

int presjek2(int *a, int *b, int **p) {
    int i = 0, j = 0, k = 0;
    while (i < NA && j < NB) {
        if (a[i] == b[j]) {
            *p = realloc(*p, (k + 1) * sizeof(int));
            (*p)[k] = a[i];
            ++ k; ++ i; ++ j;
        } else if (a[i] < b[j]) {
            ++ i; } else {
            ++ j; } }

    return k; }

```

```

void InsertionSort(int a[], int n) {
    int j,k,i;
    for(j = 1; j < n; j++) {
        k = a[j];
        i = j - 1;
        while (a[i] > k && i >= 0) {
            a[i + 1] = a[i];
            i--; }
        a[i + 1] = k; }
}

```

```

void main() {
    int a[NA] = {1,8,2,22,19,17,21,11,72,7};
    int b[NB] = {2,19,3,34,72,99,35,0,5,14,7,31,25,67,21};
    int *p = NULL, i, n;

    printf("\n Presjek nesortiranih skupova:\n");
    n = presjek1(a, b, &p);
    printf ("Presjek: ");
    for (i = 0; i < n; i++)
        printf ("%d ", p[i]);
}

```

```
printf("\ Presjek sortiranih skupova:\n");
InsertionSort(a,NA);
InsertionSort(b,NB);
printf("Polje a nakon sortiranja: ");
for (i = 0; i < NA; i ++ )
    printf ("%d ", a[i]);
printf ("\n");
printf("Polje b nakon sortiranja: ");
for (i = 0; i < NB; i ++ )
    printf ("%d ", b[i]);
printf ("\n");

n = presjek2(a, b, &p);
printf ("Presjek: ");
for (i = 0; i < n; i ++ )
    printf ("%d ", p[i]);
system("PAUSE");
}
```

Implementacija skupa pomoću bit-vektora

- Skup se prikazuje poljem bitova- svakom mogućem elementu skupa odgovara 1 bit, ako je element u skupu njegov bit sadrži vrijednost 1, inače 0
- Uzmimo za primjer skupove čiji su elementi cijeli brojevi od 0 do N
- Veličina polja za prikaz skupa u byte-ovima:
ako rabimo polje cijelih brojeva: 1 podatak = 4 byte-a = 32 bita, $M=(N+1)/32$
- Operatori između bitova:
 - & - logički "i": rezultat je 1 ako su oba bita 1
 - | - logički "inkluzivni ili": rezultat je 1 ako je bar jedan bit 1
 - ^ - logički "ekskluzivni ili": rezultat je 1 ako je točno jedan bit 1 (tj. ako su različiti)
 - << i >> - operatori pomaka određenog broja bitova
- Zadatak: napisati program kojim se pomoću bit-vektora prikažu 2 skupa cijelih brojeva čiji elementi se nalaze između 0 i N (određuje se na početku izvođenja programa). Napisati funkcije koje nađu presjek, uniju i razliku ta 2 skupa. Ispisati sve skupove.

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef unsigned int ele;
int seleb=sizeof(ele), sele, no_cel;

void nul_skup(ele *S){
    int ind;
    for (ind=0;ind<no_cel;ind++) {
        S[ind]=0;}
}

void puni_skup(ele *S, int noel, ele maxel){
    int ind,no_by,no_bi;
    ele novi;
    ind=0;
    do {
        novi=(ele) maxel *(((float) rand()/(RAND_MAX+1)));
        no_by=novi/sele;
        no_bi=novi%sele;
        if ((S[no_by] & 1<<no_bi)==0) {
            S[no_by]=S[no_by] | 1<<no_bi;
            ind++; }
    } while(ind<noel);
}

```

```

void ispis_skup(ele *S){
    int ind,ibit;
    printf("Ispis elemenata skupa:\n");
    for (ind=0;ind<no_cel;ind++) {
        for (ibit=0;ibit<sele;ibit++) {
            if (S[ind] & 1<<ibit) printf("%d ",ind*sele+ibit);
        }
        printf("\n"); }
}

```

```

ele * presjek(ele *A, ele *B){
    ele *P;
    int ind;
    P=malloc(seleb*no_cel);
    for (ind=0;ind<no_cel;ind++)
        P[ind]=A[ind] & B[ind];
    return &P[0];
}

```

```

ele * unija(ele *A, ele *B){
    ele *P;
    int ind;
    P=malloc(seleb*no_cel);
    for (ind=0;ind<no_cel;ind++)
        P[ind]=A[ind] | B[ind];
    return &P[0];
}

```

```

ele * razlika(ele *A, ele *B){
    ele *P, *R;
    int ind;
    P=malloc(seleb*no_cel);
    R=malloc(seleb*no_cel);
    for (ind=0;ind<no_cel;ind++) {
        R[ind]=B[ind] ^ 0xffffffff;
        P[ind]=A[ind] & R[ind];}
    return &P[0];
}

```

```

int main(){
    int ind, noa, nob;
    ele maxel, *A, *B, *C, *D, *E,*F;
    sele=seleb*8;
    printf("Unesite vrijednost maksimalnog elementa skupova\n");
    scanf("%d",&maxel);
    no_cel=maxel/(sele)+1;
    A=malloc(seleb*no_cel);
    B=malloc(seleb*no_cel);
    printf("Koliko elemenata ima skup A\n");
    scanf("%d",&noa);
    printf("Koliko elemenata ima skup B\n");
    scanf("%d",&nob);
    nul_skup(A,noa);
    nul_skup(B,nob);
}

```



```
srand(time(NULL));
srand(time(NULL)/rand());
printf("Skup A:\n");
puni_skup(A,noa,maxel);
ispis_skup(A);
printf("Skup B:\n");
puni_skup(B,nob,maxel);
ispis_skup(B);
C=presjek(A,B);
printf("Presjek skupova:\n");
ispis_skup(C);
D=unija(A,B);
printf("Unija skupova:\n");
ispis_skup(D);
E=razlika(A,B);
printf("Razlika A bez B:\n");
ispis_skup(E);
F=razlika(B,A);
printf("Razlika B bez A:\n");
ispis_skup(F);

system("PAUSE");
return 0;
}
```