Programozás C nyelven (11. ELŐADÁS)

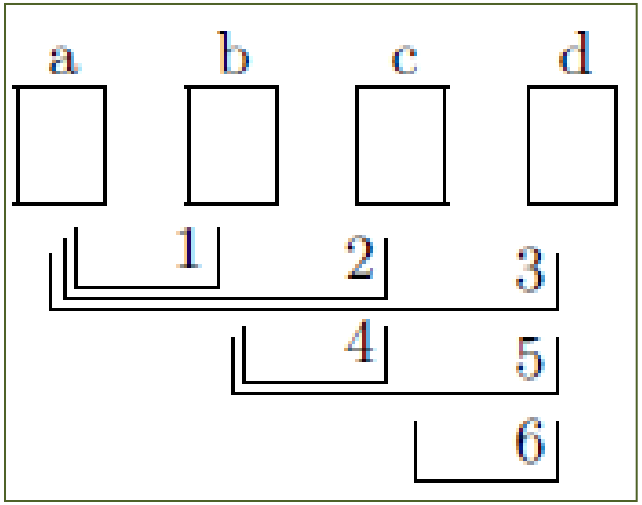
Sapientia EMTE

2015-16

Rendezés `qsort`-al

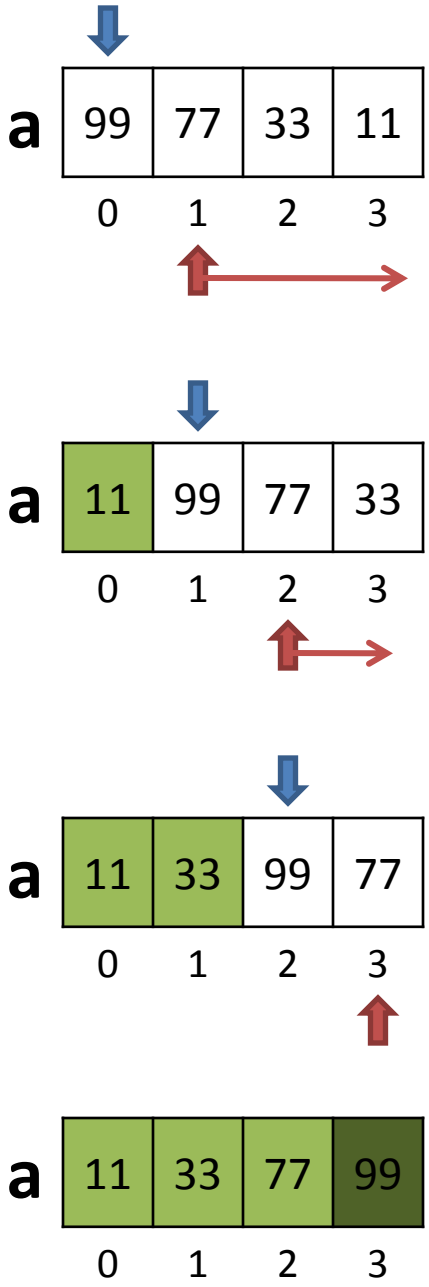
```
int double_cmp ( const void *, const void * );
int main(){
    double a[] = {5.5,4.4,3.3,2.2,1.1};
    int n = sizeof(a) / sizeof(a[0]);
    . . .
    qsort ( a, n, sizef(double), double_cmp );
    . . .
}
int double_cmp ( const void *p1, const void *p2 ){
    double *q1 = (double *)p1;
    double *q2 = (double *)p2;
    if ( *q1 < *q2 ) { return -1; }
    else if ( *q1 > *q2 ) { return 1; }
    else { return 0; }
}
```

Saját rendezési algoritmusok
KIVÁLASZTÁSOS - rendezés



algo-rythmics.ms.sapientia.ro

Saját rendezési algoritmusok
KIVÁLASZTÁSOS - rendezés




```
void kivalasztasos_rendezes(int *a, int n){
        int i, j;
        for(i = 0 ; i < n-1 ; ++i){
                for(j = i + 1 ; j < n ; ++j){
                        if(a[i] > a[j]){
                                int v = a[i];
                                a[i] = a[j];
                                a[j] = v;
                        }
                }
        }
}
```

KOMPLEXITÁS

Legrosszabb eset (csökkenő sorozat) [$O(n^2)$]:

$(n-1)+(n-2)+\ldots+1 = n(n-1)/2$ összehasonlítás + csere

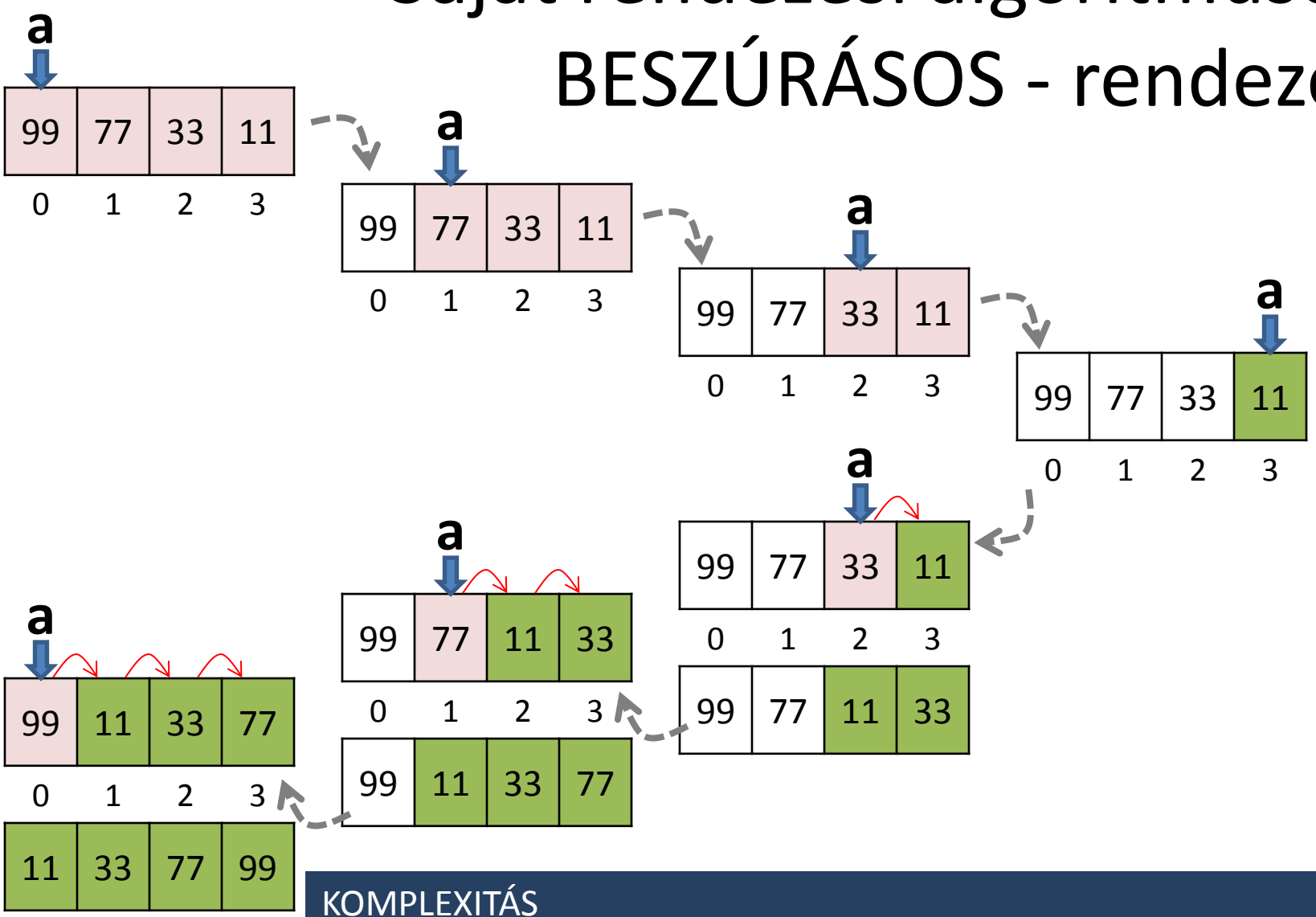
Legjobb eset (növekvő sorozat) [$O(n^2)$] :

$(n-1)+(n-2)+\ldots+1 = n(n-1)/2$ összehasonlítás

Saját rendezési algoritmusok
BESZÚRÁSOS - rendezés

algo-rythmics.ms.sapientia.ro

Saját rendezési algoritmusok
BESZÚRÁSOS - rendezés



KOMPLEXITÁS

Legrosszabb eset [$O(n^2)$]: $1+2+\ldots+(n-1)= n(n-1)/2$ összehasonlítás + csere

Legjobb eset [$O(n)$] : $(n-1)$ összehasonlítás

Saját rendezési algoritmusok
BESZÚRÁSOS - rendezés

```
void beszurasos_rendezes(int *a, int n){ //a[0..n-1] rendezése
       if(n == 1) { return; }
       beszurasos_rendezes(a+1, n-1); //a[1..n-1] rendezése
       beszuras(a,n); //az a[0] elem beszurása
}                     //az a[1..n-1] rendezett szakaszba


void beszuras(int *a, int n){
       if(n == 1) { return; }
       if(a[0] < a[1]){ return; } //*a < *(a+1)
       int v = a[0]; a[0] = a[1]; a[1] = v;
       beszuras(a+1,n-1);
}
```

Saját rendezési algoritmusok
BUBORÉKOS - rendezés

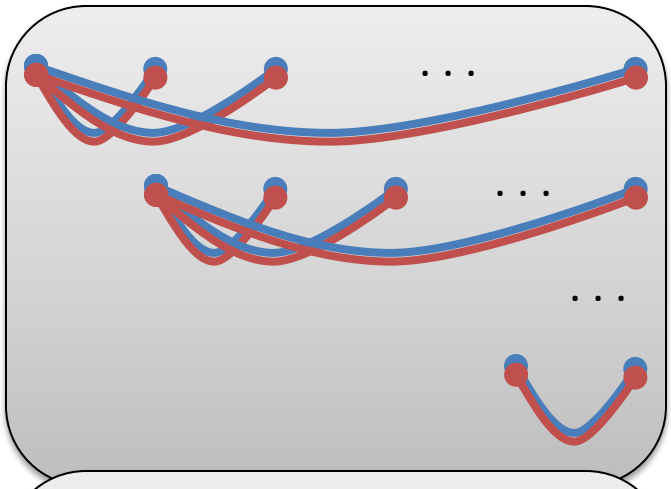
algo-rythmics.ms.sapientia.ro

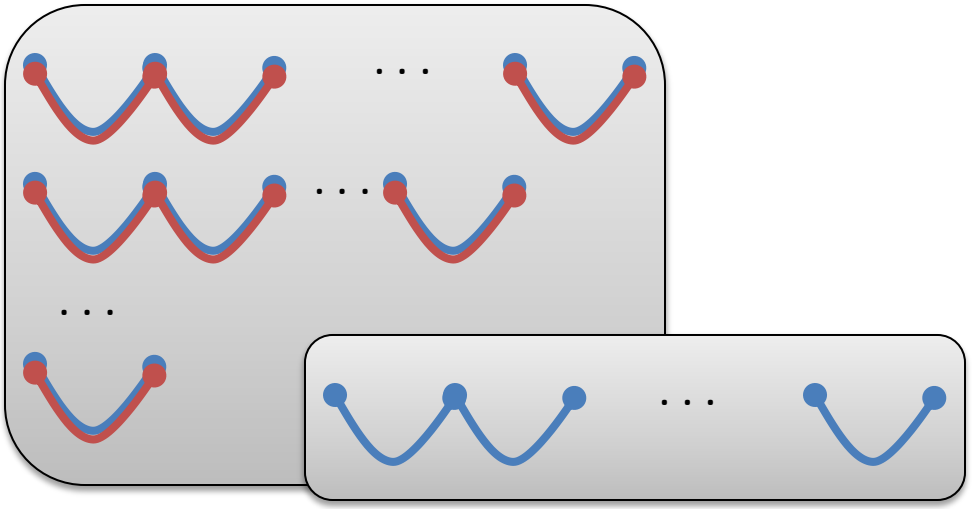
Saját rendezési algoritmusok
SHELL - rendezés

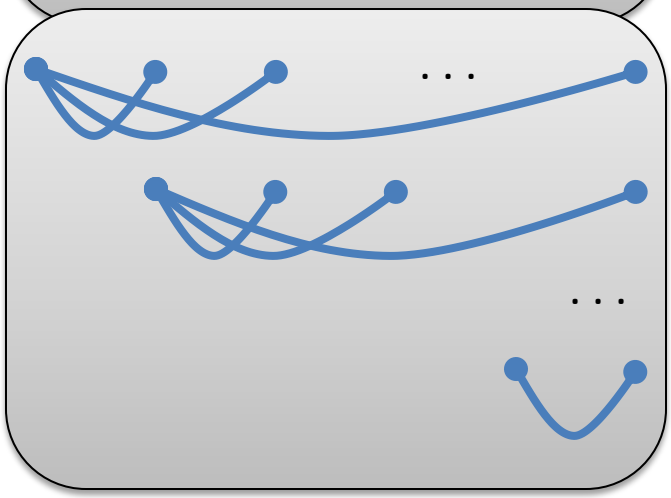
algo-rythmics.ms.sapientia.ro

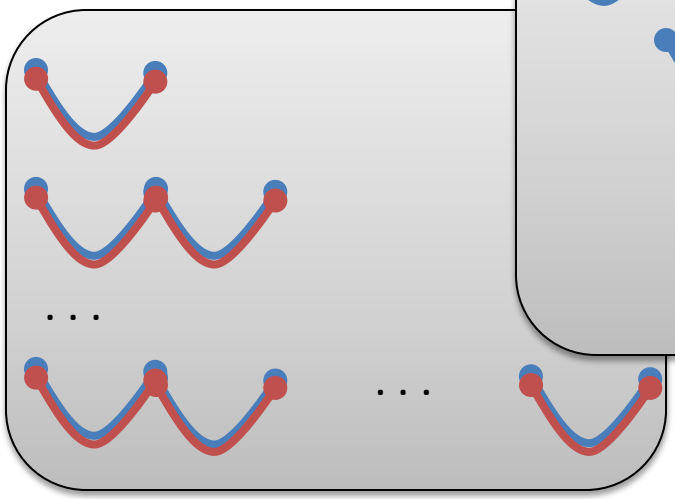
Összehasonlítás

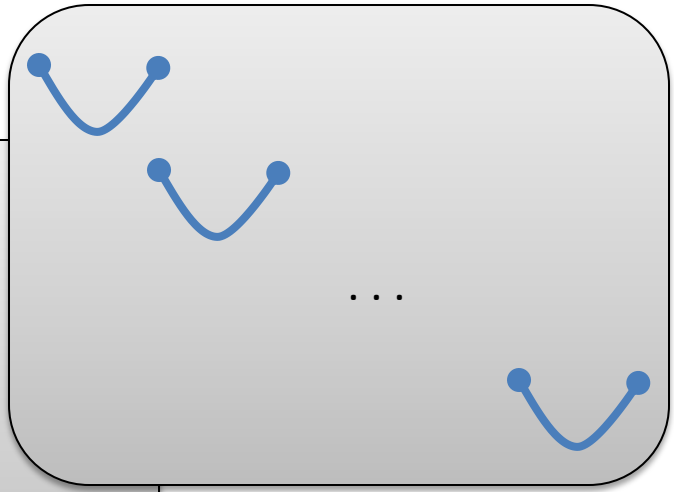
Összehasonlítás
+
Csere



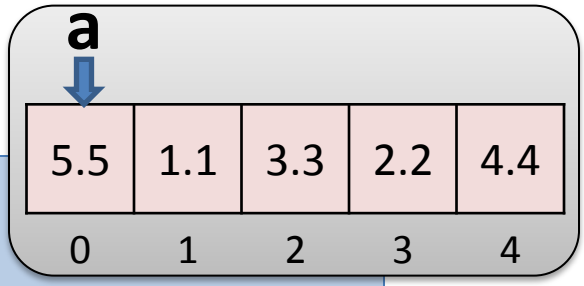






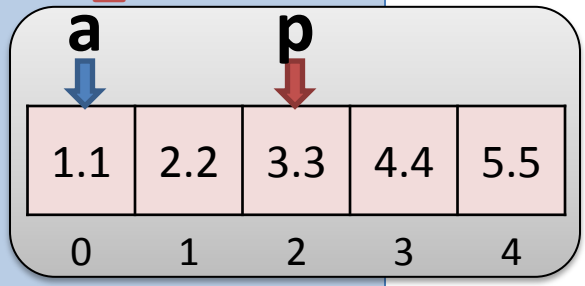


Keresés `bsearch`-el

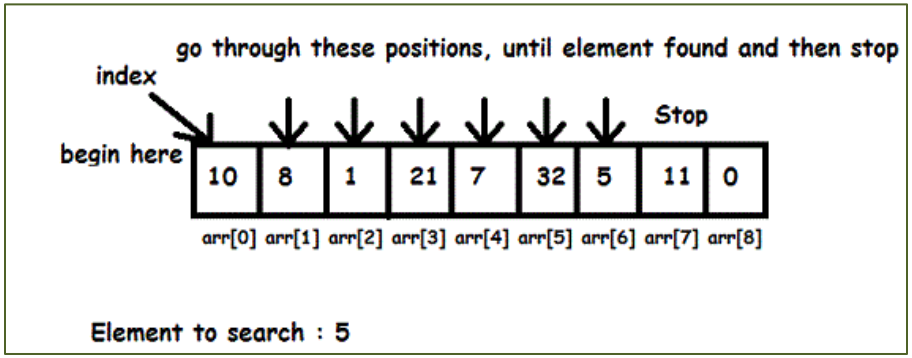


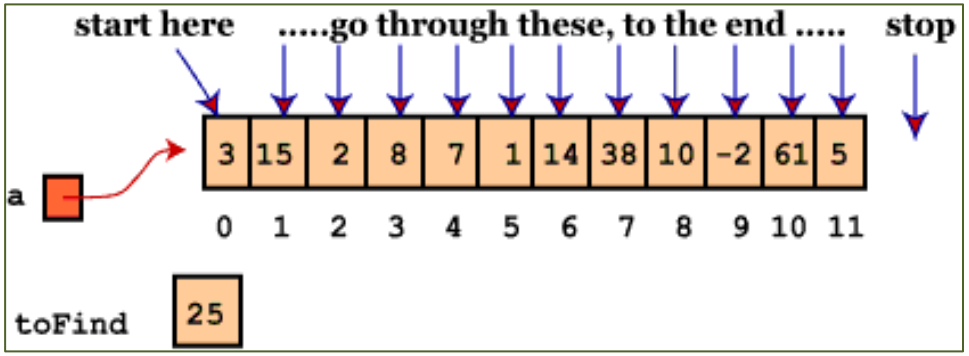

```
int double_cmp (const void *, const void *);
int main(){
    double a[] = {5.5,1.1,3.3,2.2,4.4}, x = 3.3, *p;
    int n = sizeof(a) / sizeof(a[0]);
    . . .
    qsort ( a, n, sizef(double), double_cmp );
    p = bsearch ( &x, a, n, sizef(double), double_cmp );
    if ( p ) { cout << p - a; }
    . . .
}

int double_cmp ( const void *p1, const void *p2 ){
    double *q1 = (double *)p1; double *q2 = (double *)p2;
    if ( *q1 < *q2 ) { return -1; }
    else if ( *q1 > *q2 ) { return 1; }
    else { return 0; }
}
```



Lineáris keresés






```
int linearis_kerese(int x, int *a, int n){
        int i;
        for( i = ... ; i < ... ; ++i ){
                if(...){
                        return i;
                }
        }
        return -1;
}
```

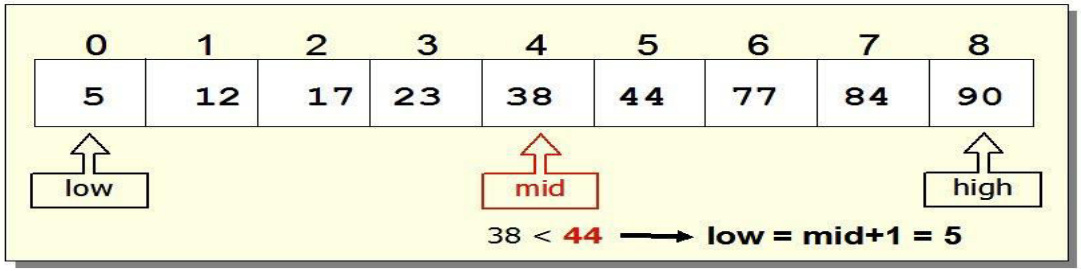
?

Bináris keresés

	low	high	mid
#1	0	8	4

search(44)

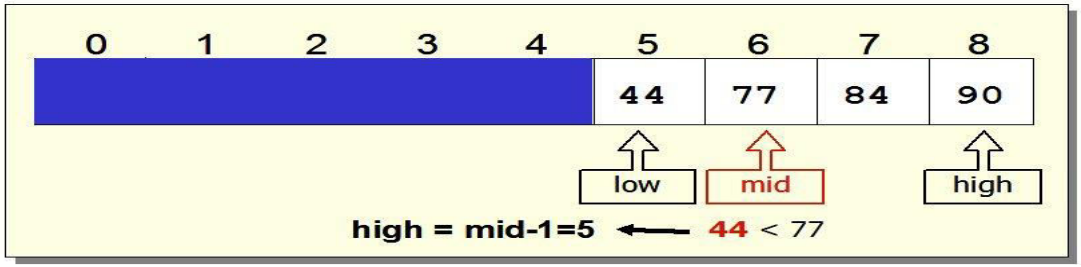
$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$



	low	high	mid
#1	0	8	4
#2	5	8	6

search(44)

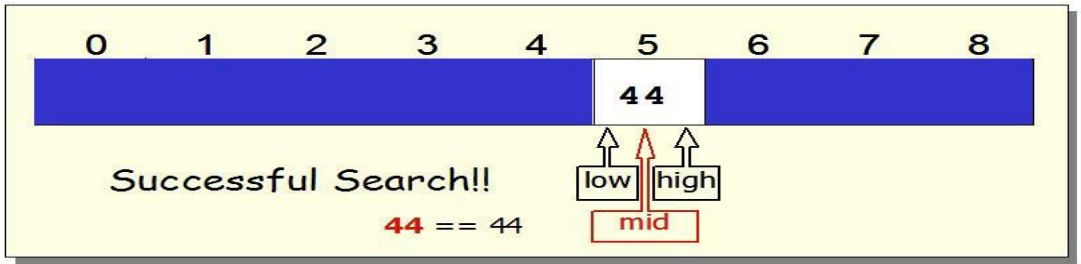
$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$



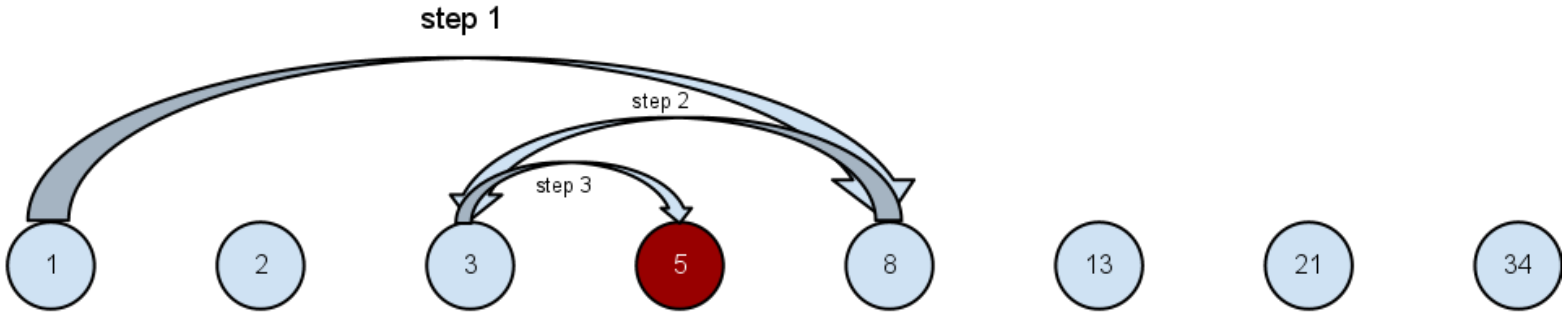
	low	high	mid
#1	0	8	4
#2	5	8	6
#3	5	5	5

search(44)

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$



Bináris keresés




```
void binaris_kerese(int x, int *a, int low, int high){
    int mid;
    if (low > high) { return -1; }
    mid = (low + high) /2;
    if (a[mid] == x) { return mid; }
    if (a[mid] < x) { return binaris_kerese(...); }
    if (a[mid] > x) { return binaris_kerese(...); }
}
```

?