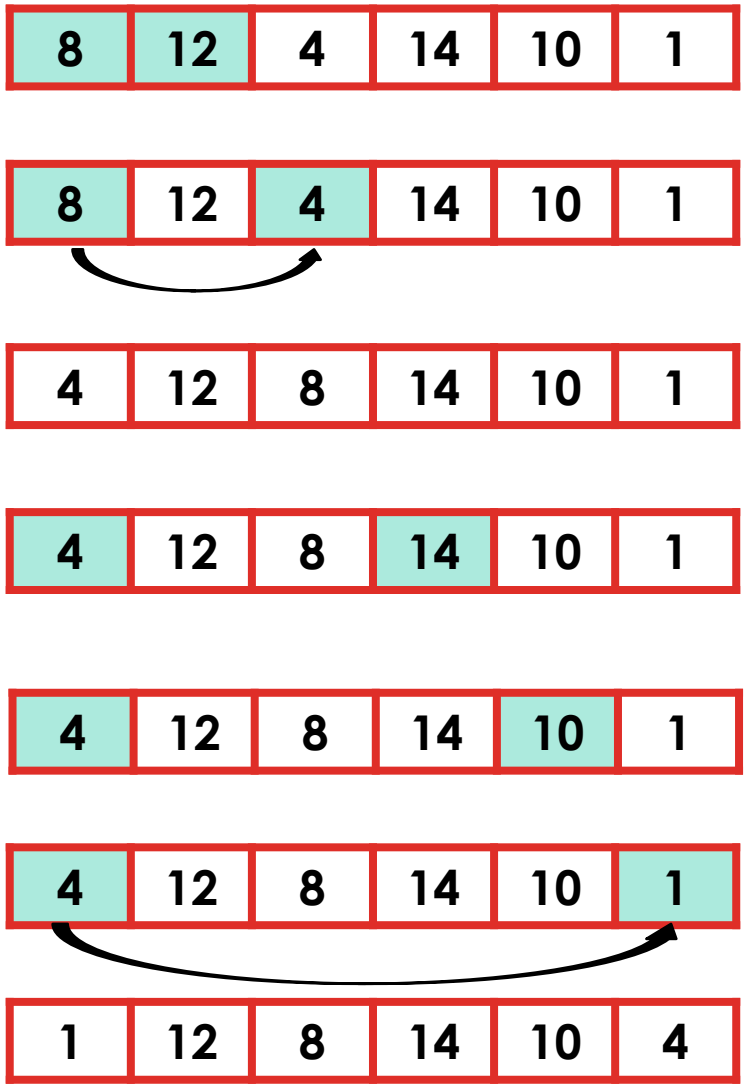
ALGORITMI SORTIRANJA

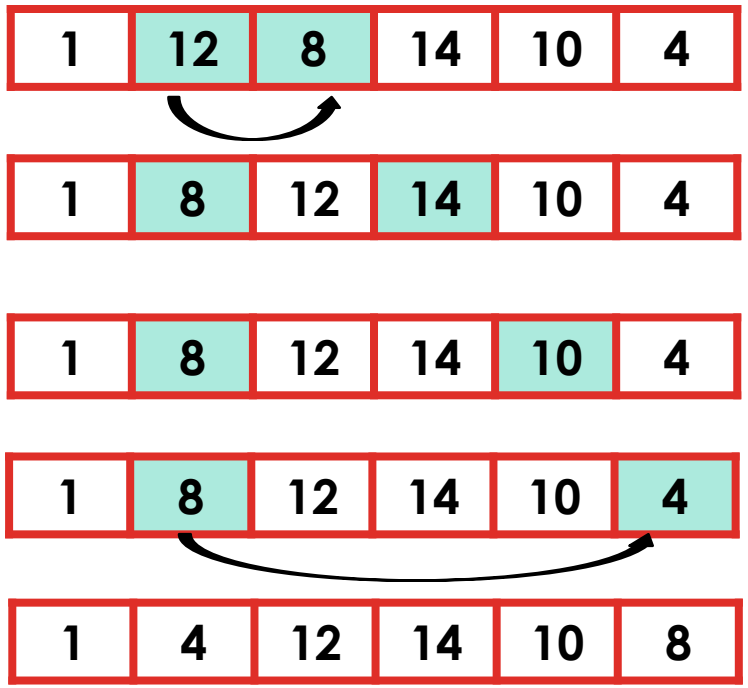
EXCHANGE SORT

EXCHANGE SORT (SORTIRANJE ZAMJENOM)

- često neke podatke treba sortirati
- jedan jednostavan način soritranja elemenata liste
- na prvo mjesto postavljamo najmanji element, na drugo mjesto drugi po veličini itd.
- zamjena se odvija tako da prvi element uspoređujemo sa svima i kad naiđemo na manji element od prvog onda ih zamijenimo
- nakon prvog elementa na isti način nastavljamo na drugi pa treći...

PRIMJER (EXCHANGE)





Ponavljamo do kraja...

EXCHANGE SORT

```
# exchange sort (od manjeg prema većem)
L = [5, 4, 9, 1, 8, 2]
for i in range(len(L)-1):  #gledamo od prvog elementa do predzadnjeg
    for j in range(i+1, len(L)): #uspoređujemo s elementima poslije i-tog do zadnjeg
        if L[i]>L[j]:     #usporedba
            L[i], L[j] = L[j], L[i] #zamjena elemenata

print(L)
```

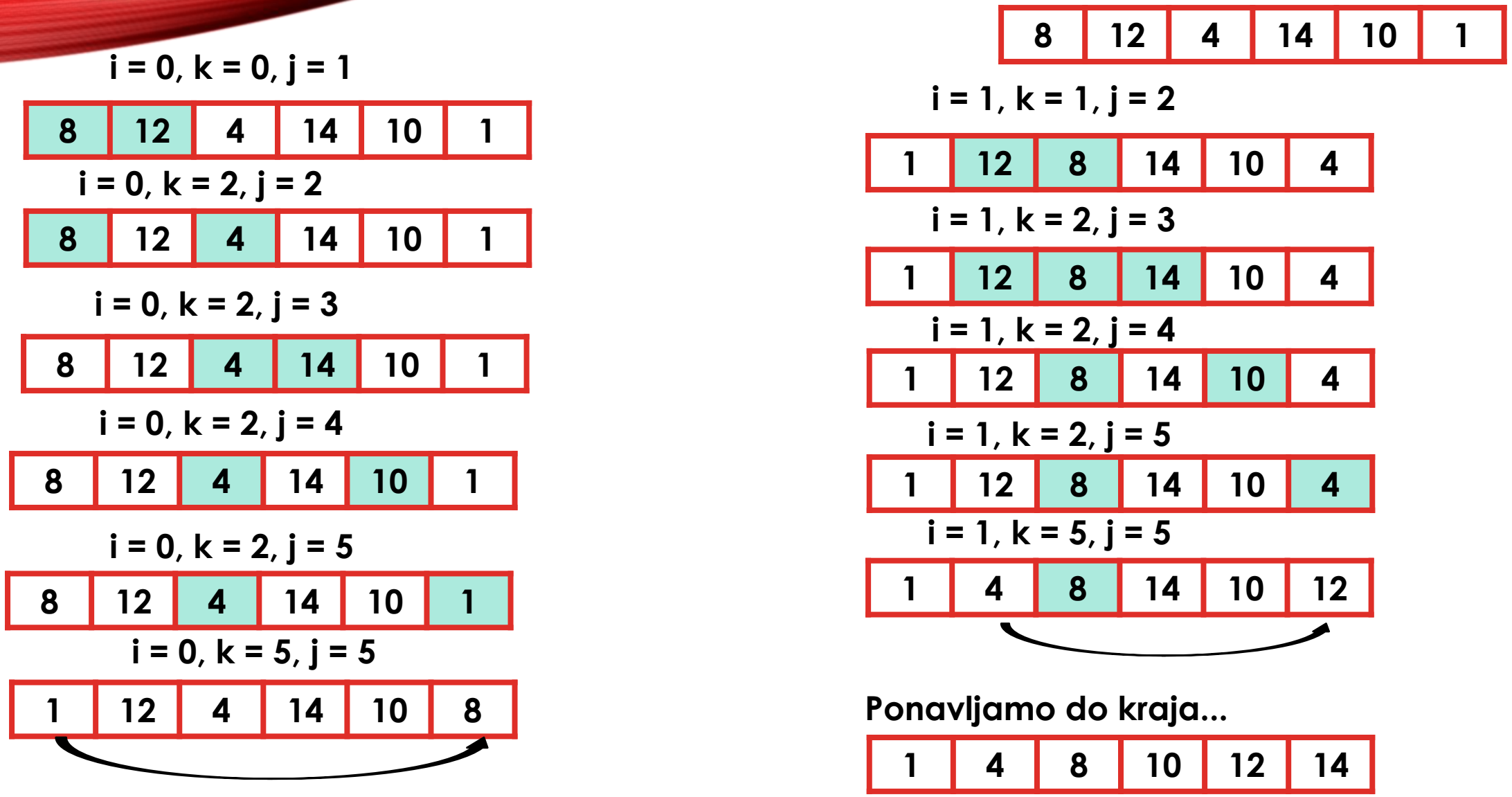
ALGORITMI SORTIRANJA

SELECTION SORT

SELECTION SORT (SORTIRANJE IZBOROM)

- slično je sortiranju zamjenom elemenata
- razlika je da se samo pamti indeks najmanjeg elementa i zamjena se tek radi na kraju for petlje

PRIMJER (SELECTION)



SELECTION SORT

```
# selection sort (od manjeg prema većem)
L = [5, 4, 9, 1, 8, 2]
for i in range(len(L)-1):  #gledamo od prvog elementa do predzadnjeg
    k = i       #pokazivač nam je na početku na i-tom
    for j in range(i+1, len(L)): #uspoređujemo k-ti element s elementima poslije i-tog do zadnjeg
        if L[k]>L[j]:     #usporedba
            k = j #promjena pokazivača (nismo mijenjali poredak elemenata liste)
    if k != i:
        L[i], L[k] = L[k], L[i]

print(L)
```

ZADACI

1. Napiši program koji će zadanu listu, L = [7, 2, 4, 9, 11, 0] sortirati silazno (od većeg prema manjem):
    a) pomoću exchange sorta
    b) pomoću selection sorta
2. Napiši program koji će zadanu listu:

L = [‘maja’, ‘ana’, ‘petra’, ‘zoran’, ‘bojan’, ‘karla’] sortirati od slova a do slova z.

ALGORITMI SORTIRANJA

BUBBLE SORT

BUBBLE SORT (SORTIRANJE ZAMJENOM SUSJEDNIH ELEMENATA)

- opet slično je sortiranju zamjenom elemenata
- za razliku od njega najveći se elementi redaju na kraju liste
- na početku pretpostavimo da moramo zamijeniti neke elemente u listi te ćemo ponavljati taj program sve dok moramo mijenjati elemente
- u prvom prolazu na kraj liste doći najveći element, u drugom prolazu će na kraj liste doći drugi po veličini element itd. te ćemo u svakom sljedećem prolazu sortirati za jedan kraću listu

PRIMJER (BUBBLE)

8	12	4	14	10	1

i = 0, i+1 = 1, zam = 0

8	12	4	14	10	1

i = 1, i+1=2, zam = 1

8	12	4	14	10	1

i = 1, i+1=2, zam = 1

8	4	12	14	10	1

i = 2, i+1=3, zam = 1

8	4	12	14	10	1

i = 3, i+1=4, zam = 1

8	4	12	14	10	1

i = 3, i+1=4, zam=1

8	4	12	10	14	1

i = 4, i+1=5, zam=1

8	4	12	10	14	1

8	4	12	10	1	14

Više ne gledamo do zadnjeg elementa nego predzadnjeg jer je zadnji najveći

8	4	12	10	1	14

Ponavljamo do kraja...

1	4	8	10	12	14

BUBBLE SORT

```
L = [1, 5, 7, 6, 2]
k = 1    #do kojeg elementa uspoređujemo (len -1)
zamjena = True    #pretpostavino da moramo jednom mijenjati
while zamjena:
    zamjena = False
    for i in range(len(lista) - k): #ne gledamo svaki puta sve elemente(zadnji us sortirani)
        if lista[i] > lista[i + 1]: #mijenjamo ukoliko je potrebno
            lista[i], lista[i + 1] = lista[i + 1], lista[i]
            zamjene = True
    k += 1
```

ALGORITMI SORTIRANJA

INSERTION SORT

INSERTION SORT (SORTIRANJE UMETANJEM ELEMENATA)

- opet slično kao i prethodni algoritmi imamo dva dijela liste – sortirani i nesortirani
- na početku imamo sortirani dio liste samo od prvog elementa
- sortiranje se provodi tako da on u svakom prolazu uzima samo jedan element i pomiče ga ulijevo dok ne dođe na pravo mjesto sortirane liste

PRIMJER (INSERTION)

i = 1, j = 1, j-1 = 0

8	12	4	14	10	1

i = 2, j=2, j-1 = 1

8	12	4	14	10	1

i = 2, j=1, j-1 = 0

8	4	12	14	10	1

i = 3, j=3, j-1 = 2

4	8	12	14	10	1

i = 4, j=4, j-1 = 3

4	8	12	14	10	1

i = 4, j=3, j-1=2

4	8	12	10	14	1

i = 4, j=2, j-1=1

4	8	10	12	14	1

i **je prvi element nesotriranog dijela, a *j* je element s kojim uspoređujem o prethodni**

8	12	4	14	10	1

4	8	10	12	14	1

Ponavljamo do kraja...

1	4	8	10	12	14

INSERTION SORT

```
L = [4, 2, 6, 10, 8, 7]
for i in range(1, len(L)): #i nam ide od 1 do duljine liste
    j = i #j postavljamo na i
    while j > 0 and L[j] < L[j - 1]: #ako mijenjamo elemente smanjujemo j
        L[j - 1], L[j] = L[j], L[j - 1]
        j -= 1
print(L)
```

ALGORITMI SORTIRANJA

MERGE SORT

PRIMJER PRIJE MERGE SORTA

Napišite program ili funkciju čiji će parametri biti dvije sortirane liste, a ona će vraćati sortiranu listu koja se dobiva spajanjem dviju zadanih lista. (merge)

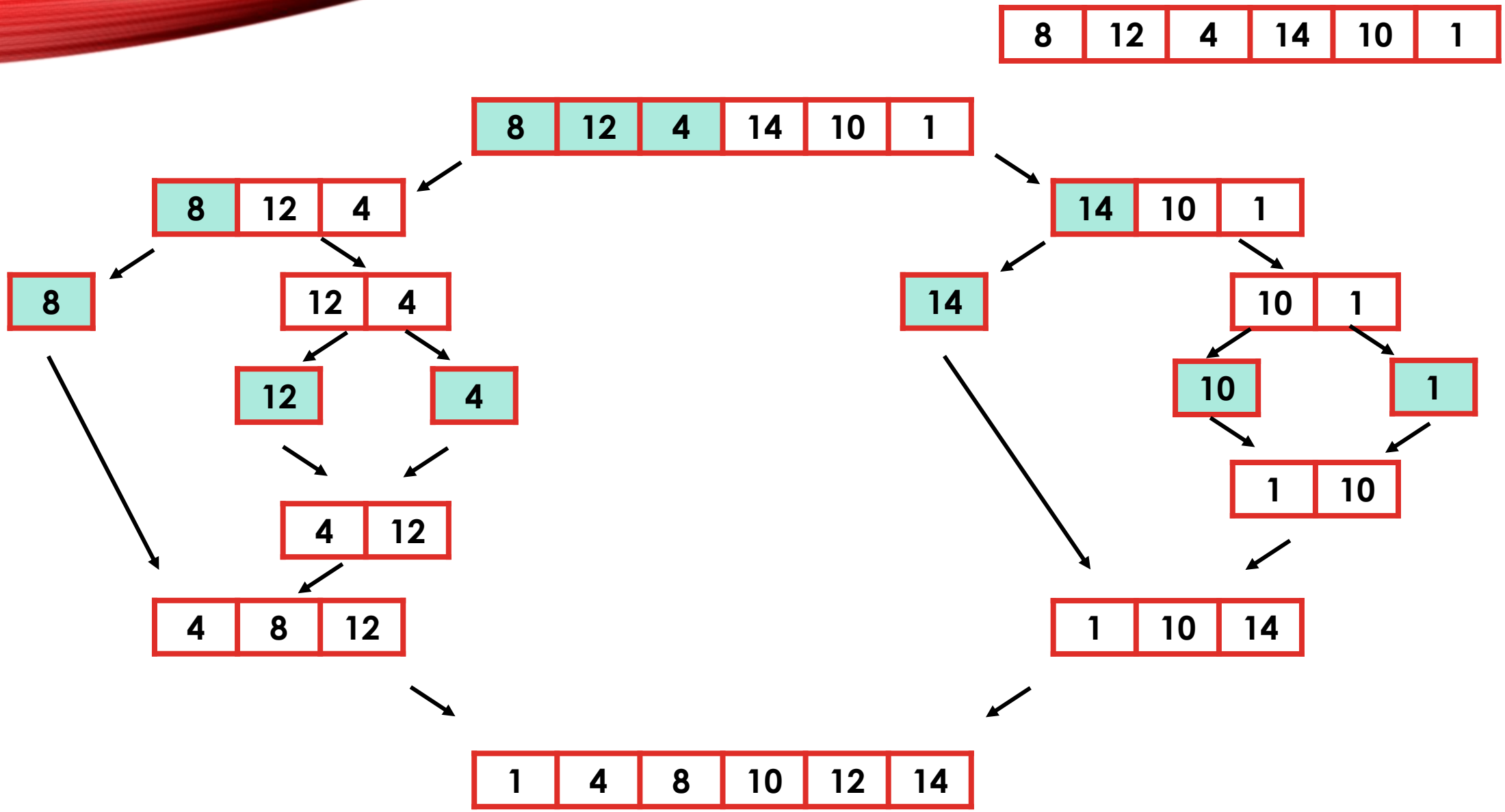
RJEŠENJE PRIMJERA PRIJE MERGE SORTA

```
def merge(a, b):
    i, j = 0, 0   #i je indeks za a listu, a j za b listu
    t = [0] * (len(a) + len(b)) #konačna lista je duljine len(a)+len(b)
    for k in range(len(a) + len(b)):
        if i < len(a) and j < len(b):
            if a[i] < b[j]:
                t[k] = a[i]
                i += 1
            else:
                t[k] = b[j]
                j += 1
        elif i < len(a):
            t[k] = a[i]
            i += 1
        else:
            t[k] = b[j]
            j += 1
    return t
a = [2, 4, 8]
b = [1, 5, 7, 9]
L = merge(a, b)
print(L)
```

MERGE SORT (SORTIRANJE SJEDINJAVANJEM)

- načelo dijeljenja većeg na manje dijelove te rješavanje po dijelovima može znatno ubrzati posao
- početnu listu podijelimo na dvije jednako dugačke podliste
- podliste treba sortirati i postupkom sjedinjavanja objediniti u jednu
- sortiranje podlista treba obaviti **rekurzivno** na isti način

PRIMJER (MERGE)



MERGE SORT

```
L = [4, 2, 6, 10, 8, 7]
def sort_merge(lista):
    if len(lista) == 1:
        return lista
    m = len(lista) // 2
    lijevo = lista[: m]
    desno = lista[m:]
    sort_merge(lijevo)
    sort_merge(desno)
    i, j = 0, 0
    for k in range(len(lijevo) + len(desno)):
        if i < len(lijevo) and j < len(desno):
            if lijevo[i] < desno[j]:
                lista[k] = lijevo[i]
                i += 1
            else:
                lista[k] = desno[j]
                j += 1
        elif i < len(lijevo):
            lista[k] = lijevo[i]
            i += 1
        else:
            lista[k] = desno[j]
            j += 1
    return lista
sort_merge(L)
print(L)
```

ALGORITMI SORTIRANJA

QUICK SORT

QUICK SORT (BRZO SORTIRANJE PO HOAREU)

- također je rekurzivan kao i merge sort samo se početna lista ne dijeli na jednako velike podliste
- na proizvoljni način odabire se jedan od elemenata za ključni element ključni_el i kreiraju se dvije podliste (podlista onih elemenata koji su manji ili jednaki ključnom elementu i podlista onih elemenata koji su veći od ključnog elementa) i one se dalje rekurzivno sortiraju
- imamo dva pokazivača: prvi pokazivač (i) ide od početka liste udesno – tražit ćemo prvi element koji je veći ili jednak ključnom, a drugi pokazivač(j) ide od kraja liste ulijevo – tražit ćemo element koji je manji ili jednak ključnom.

PRIMJER (QUICK)

8	12	4	14	10	1

i = 0, j = 5, k = 2

8	12	4	14	10	1

i je na početku, a *j* na kraju

i = 0, j=5, k = 2

8	12	4	14	10	1

i = 1, j=4, k = 2

1	12	4	14	10	8

Ponavljamo do kraja...

i = 1, j=3, k = 2

1	12	4	14	10	8

i = 1, j=2, k = 2

1	12	4	14	10	8

1	4	12	14	10	8

i = 2, j=1, k = 1 – i, j idu do i<=j

1	4	12	14	10	8

1	4	8	10	12	14

ponavljamo rekurzivno na dvije liste L[0-1] i L[2-5] (indeksi)

QUICK SORT

```
L = [4, 7, 2, 14, 10, 1]
def sort_quick(lista, poc, kraj): #početni je indeks 0, a krajni len(L)-1
    sred = lista[(poc + kraj) // 2] #pivot odnosno ključni element
    i = poc
    j = kraj
    while i <= j:
        while lista[i] < sred:  #ako nemamo elemente koji su veći od pivota povećavamo i
            i += 1
        while lista[j] > sred:  #ako nemamo elemente koji su manji od pivota povećavamo j
            j -= 1
        if i <= j:
            lista[i], lista[j] = lista[j], lista[i] #mijenjamo ako imamo veći i manji element
            i += 1
            j -= 1
    if poc < j: #ako nismo provjerili sve elemente
        sort_quick(lista, poc, j)
    if kraj > i:
        sort_quick(lista, i, kraj)
    return lista
print(sort_quick(L, 0, len(L)-1))
```

ZADACI

1. Kako će izgledati elementi liste a = [4, 2, 7, 1, 8, 6] nakon prvog prolaza sortiranja ako se za sortiranje koristi:

a) metoda razmjene (engl. exchange sort)

b) sortiranje izborom (engl. selection sort)

c) mjehurićasto sortiranje (engl. bubble sort)

d) sortiranje umetanjem (engl. insertion sort)

2. Kako će izgledati elementi liste b = [7, 5, 1, 4, 8, 6, 9, 17, 12] nakon prvog prolaza sortiranja ako se za sortiranje koristi:

a) metoda razmjene (engl. exchange sort)

b) sortiranje izborom (engl. selection sort)

c) mjehurićasto sortiranje (engl. bubble sort)

d) sortiranje umetanjem (engl. insertion sort)

ZADACI

3. Kako će izgledati elementi liste L = [4, 2, 7, 11, 6, 9, 1, 15] nakon prvog prolaza sortiranja ako se za sortiranje koristi:

a) metoda razmjene (engl. exchange sort)

b) sortiranje izborom (engl. selection sort)

c) mjehurićasto sortiranje (engl. bubble sort)

d) sortiranje umetanjem (engl. insertion sort)

ZADACI

4. Zadana je lista L = [8, 9, 4, 17, 2]. Nakon prvog prolaza algoritma sortiranja lista ima sljedeći oblik:

a) L = [8, 4, 9, 2, 17]

b) L = [8, 9, 4, 17, 2]

c) L = [2, 9, 8, 17, 4].

Koji je od sljedećih algoritama korišten za sortiranje elemenata liste?

– metoda razmjene (engl. exchange sort)

– sortiranje izborom (engl. selection sort)

– mjehurićasto sortiranje (engl. bubble sort)

– sortiranje umetanjem (engl. insertion sort).

ZADACI

5. Zadana je lista a = [9, 5, 8, 17, 4, 6, 15, 3, 20, 11]. Nakon prvog prolaza algoritma sortiranja lista ima sljedeći oblik:

a) a = [3, 5, 8, 17, 4, 6, 15, 9, 20, 11]

b) a = [3, 9, 8, 17, 5, 6, 15, 4, 20, 11]

c) a = [5, 8, 9, 4, 6, 15, 3, 17, 11, 20].

Koji je od sljedećih algoritama korišten za sortiranje elemenata liste?

– metoda razmjene (engl. exchange sort)

– sortiranje izborom (engl. selection sort)

– mjehurićasto sortiranje (engl. bubble sort)

– sortiranje umetanjem (engl. insertion sort).

ZADACI

6. Napiši program koji će unositi prirodan broj n, zatim listu od n elemenata, a potom će elemente liste sortirati u obrnutom poretku (od najvećeg prema najmanjem). Napomena: Korištenje metode reverse() nije dozvoljeno. Za sortiranje elemenata koristite:

a) metodu razmjene (engl. exchange sort)

b) sortiranje umetanjem (engl. insertion sort)

7. Zadana je lista učenika u kojima su spremljene njihova imena i visine.

L = [[‘Mirna’, 176], [‘Filip’, 180],[‘Bojan’, 169], [‘Klara’, 166]].

Napišite program koji će prvo sortirati učenike s obzirom na visinu od većeg prema manjem korištenjem bubble sorta (ispisati listu), a zatim sortirajte učenike po imenima od A do Z koristeći insertion sort.

ZADACI

8. Napiši program koji će unositi listu od 7 brojeva. Program treba podijeliti listu na tri podliste tako da se u prvoj nalaze 3 elementa manja od srednjeg, u drugoj je srednji element, a u trećoj podlisti tri elementa veća od srednjeg elementa.

Ulaz: L = [4, 3, 6, -2, 70, 1, 10]

Izlaz: M = [-2, 1, 3]

S =[4]

V = [6, 10, 70]

9. Napiši program koji će unositi listu od n riječi. Program treba sortirati tu listu prema prema broju znakova (od najkraće prema najduljoj). Za sortiranje koristite sortiranje izborom (engl. selection sort)

Ulaz: L = [‘danas’, ‘dan’, ‘utorak’]

Izlaz; L = [‘dan’, ‘danas’, ‘utorak’]

ZADACI

10. Napiši program koji će unositi rečenicu (niz riječi međusobno odvojenih razmacima). Program treba ispisati riječi te rečenice sortirane prema broju znakova (od najkraće prema najduljoj). Za sortiranje koristite mjehurićasto sortiranje (engl. bubble sort)

b) sortiranje izborom (engl. selection sort)

ulaz	izlaz
ALEA IACTA EST	EST ALEA IACTA