Algoritmi e Strutture Dati

Lezione 6

www.iet.unipi.it/a.virdis

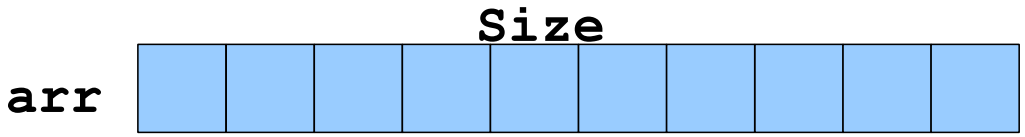
Antonio Virdis

antonio.virdis@unipi.it

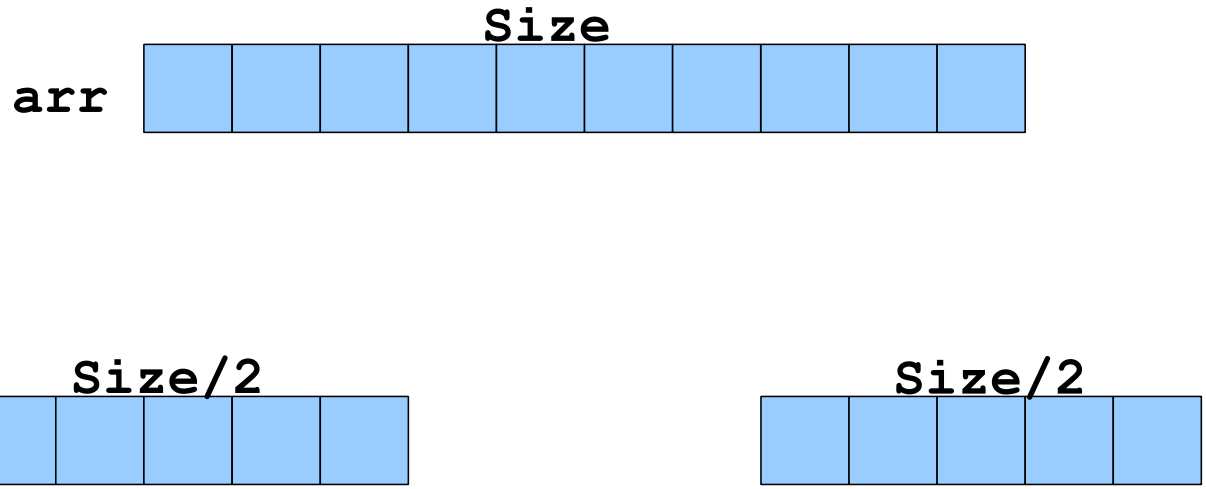
Sommario

- Merge Sort
- Ordinamento STL
- Gestione Liste
- Esercizi

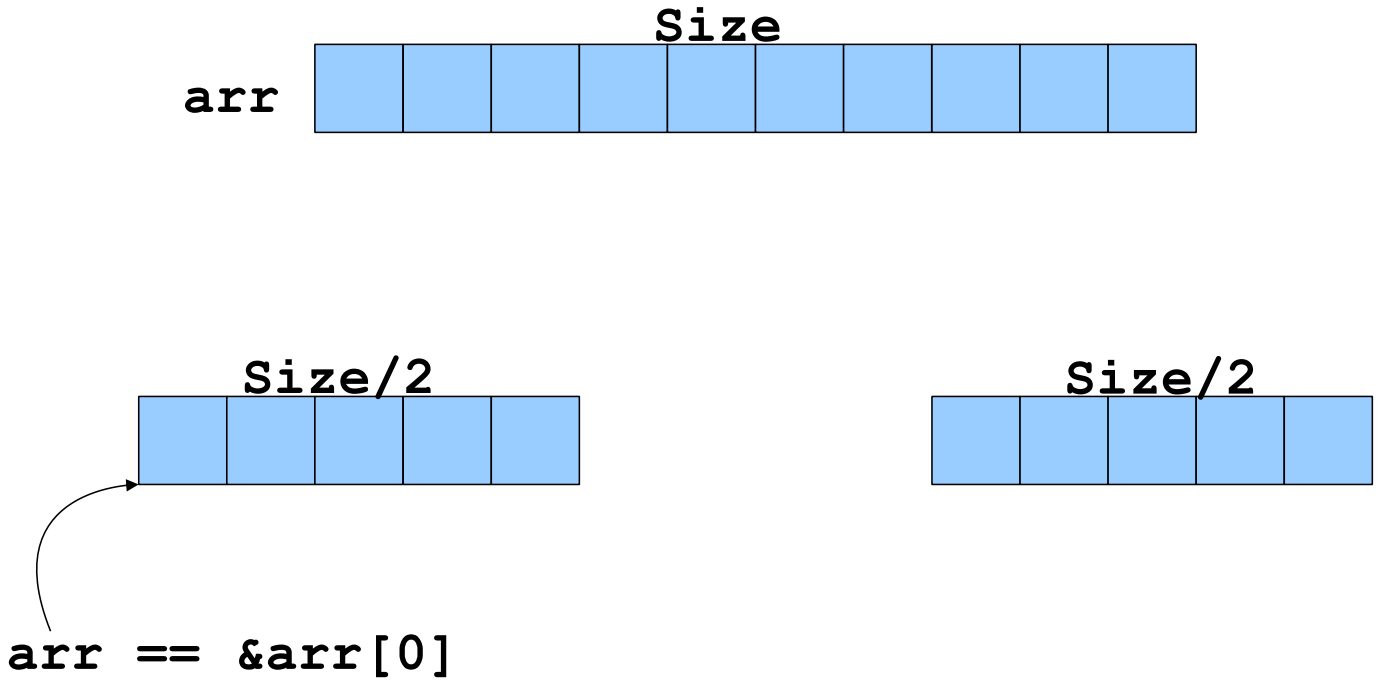
Sorting



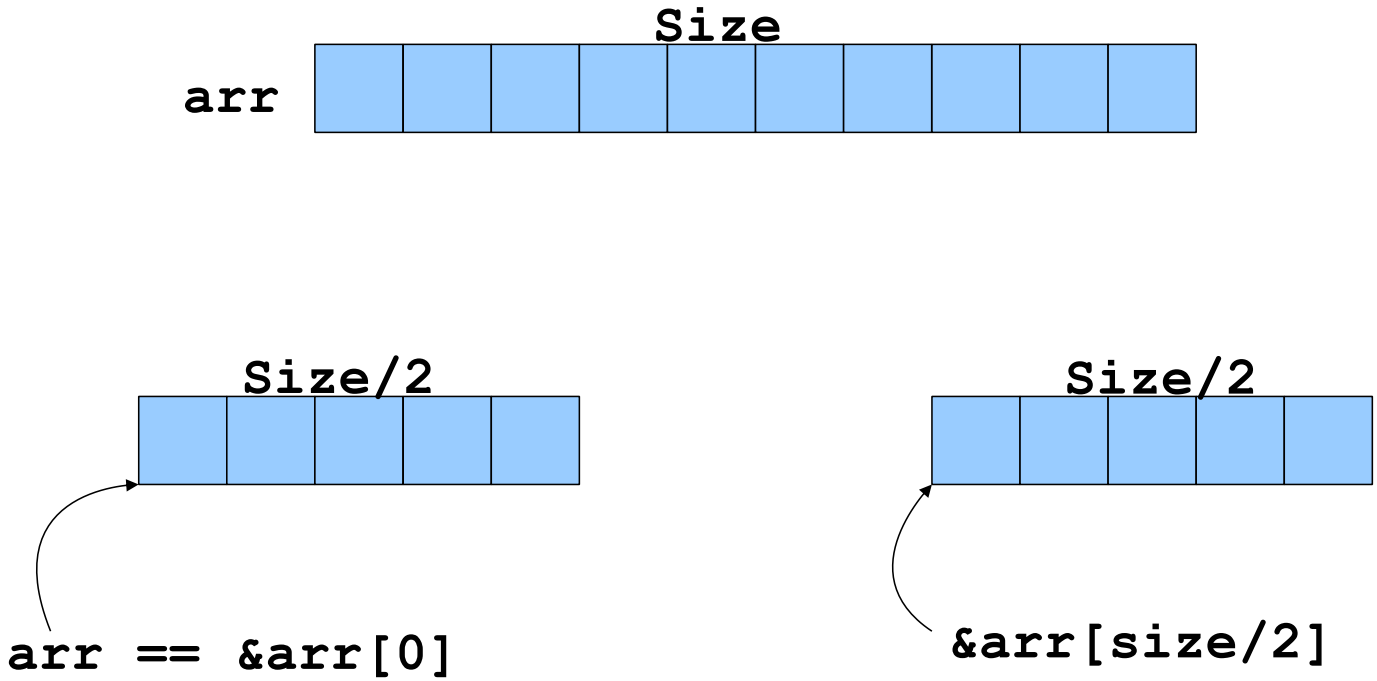
A metà



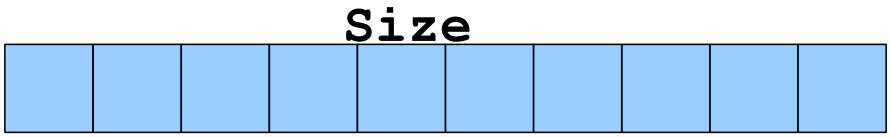
A metà

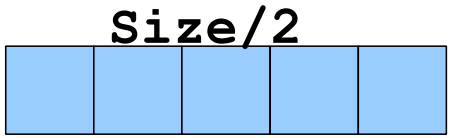


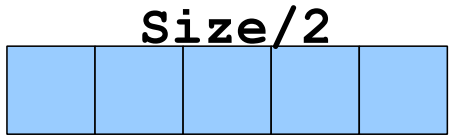
A metà



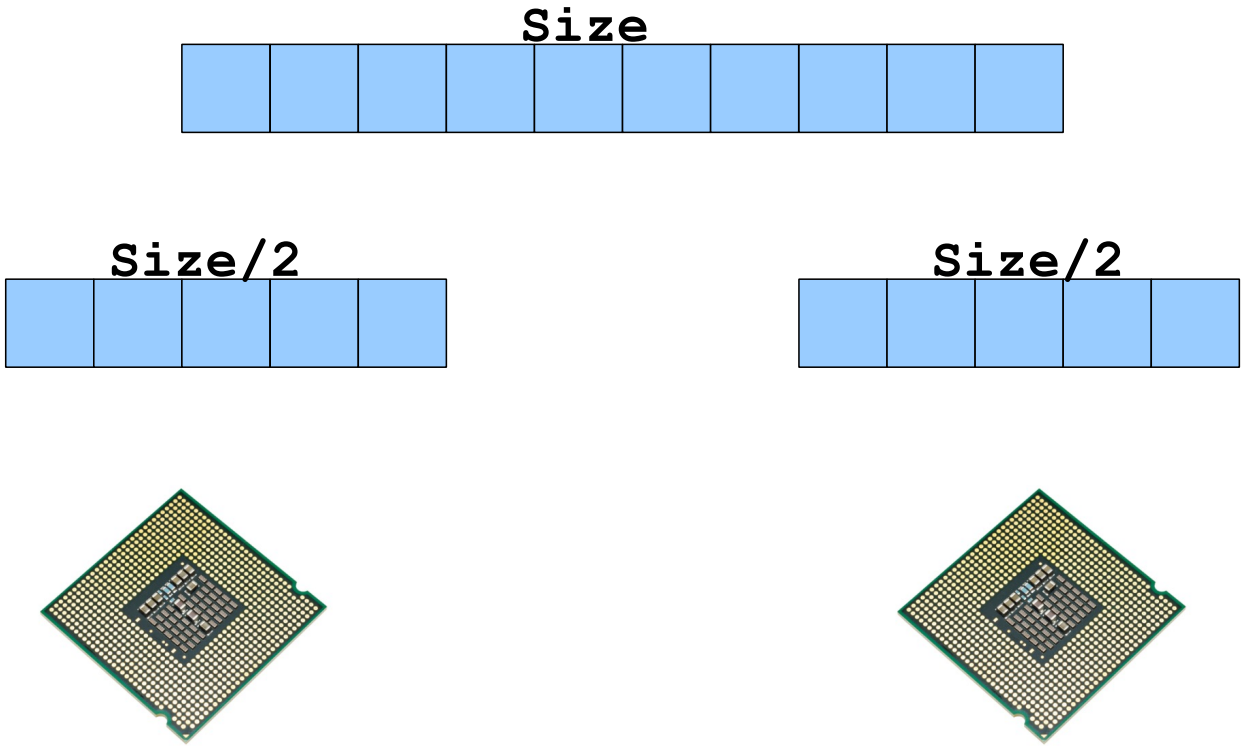
A metà



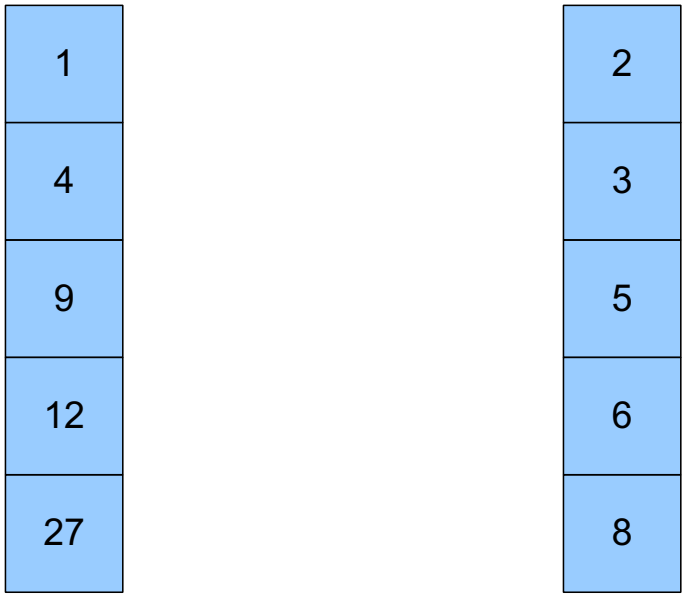




A metà



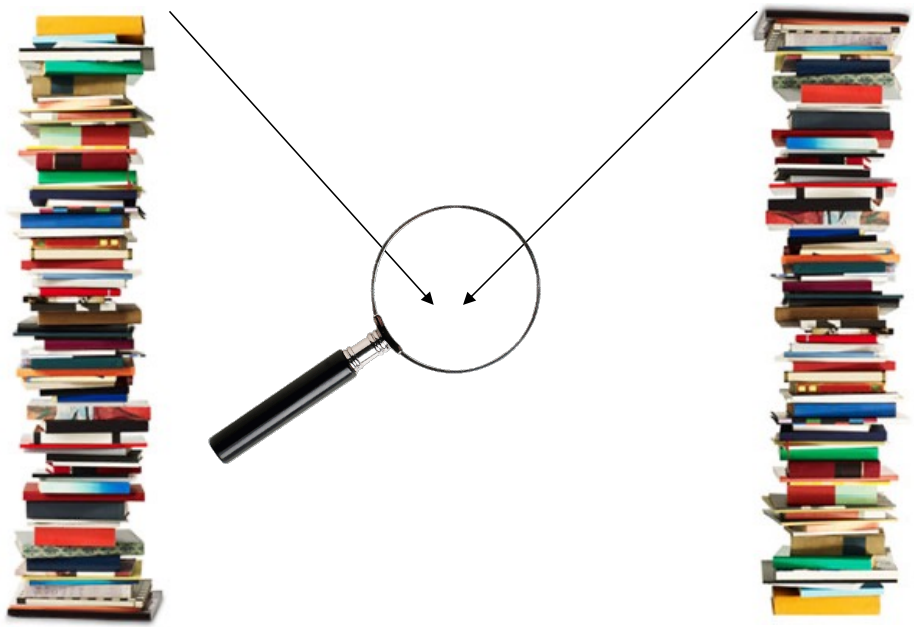

1
4
9
12
27
2
3
5
6
8


1
4
9
12
27
2
3
5
6
8

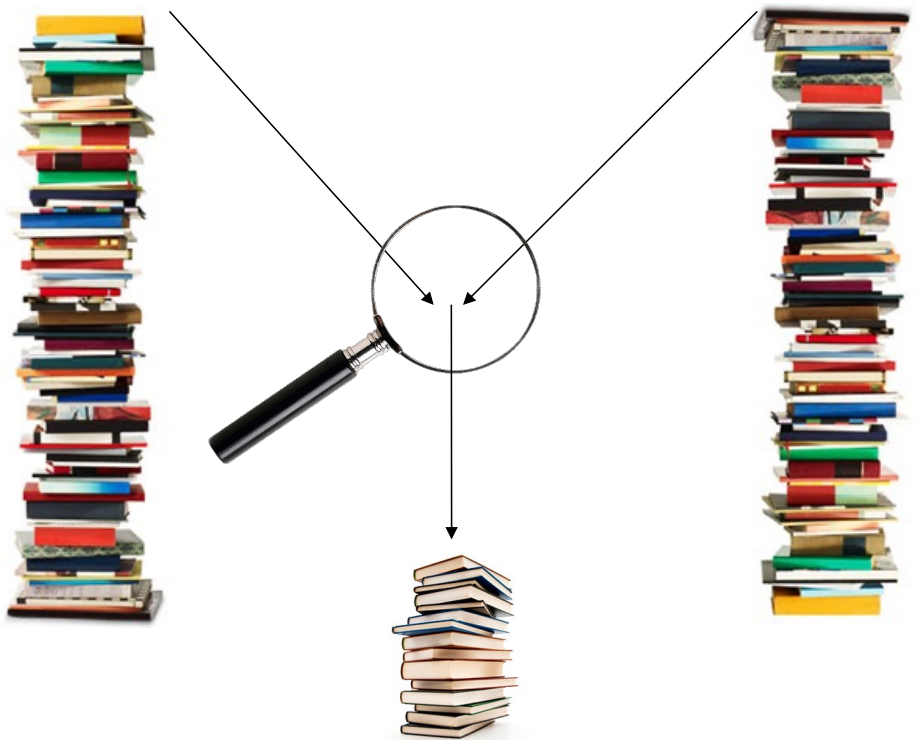
Unire gli array

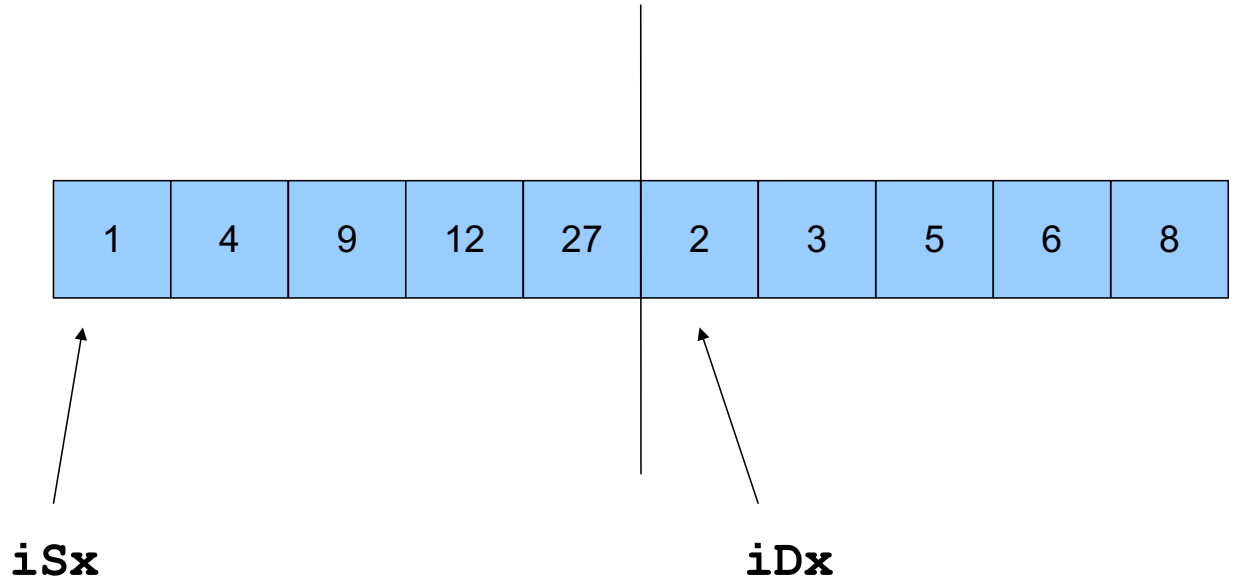
Unire gli array

Unire gli array



Unire gli array




1
4
9
12
27
2
3
5
6
8
iSx
iDx

combina

```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20  }
```

combina

```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3       // init Variabili di stato + buffer appoggio
4
5       while(1)
6       {
7
8
9
10
11
12
13
14
15
16
17      }
18
19
20  }
```

combina

```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3       // init Variabili di stato + buffer appoggio
4
5       while(1)
6       {
7           // se arr[iSx] più piccolo
8           {
9               // Inserisco arr[iSx]
10
11          }
12          // se arr[iDx] più piccolo
13          {
14              // Inserisco arr[iDx]
15
16          }
17      }
18
19
20  }
```

combina

```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3       int iSx = start , iDx = mid; // stato
4       std::vector<int> tempResult; // buffer
5       while(1)
6       {
7           if(arr[iSx] < arr[iDx])
8           {
9
10
11          }
12          else
13          {
14
15
16          }
17      }
18
19
20  }
```

combina

```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3       int iSx = start , iDx = mid; // stato
4       std::vector<int> tempResult; // buffer
5       while(1)
6       {
7           if(arr[iSx] < arr[iDx])
8           {
9               tempResult.push_back(arr[iSx++]);
10
11          }
12          else
13          {
14              tempResult.push_back(arr[iDx++]);
15
16          }
17      }
18
19
20  }
```

combina

```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3       int iSx = start , iDx = mid; // stato
4       std::vector<int> tempResult; // buffer
5       while(1)
6       {
7           if(arr[iSx] < arr[iDx])
8           {
9               tempResult.push_back(arr[iSx++]);
10
11          }
12          else
13          {
14              tempResult.push_back(arr[iDx++]);
15
16          }
17      }
18
19      // RICOPIO da buffer a arr
20  }
```

combina

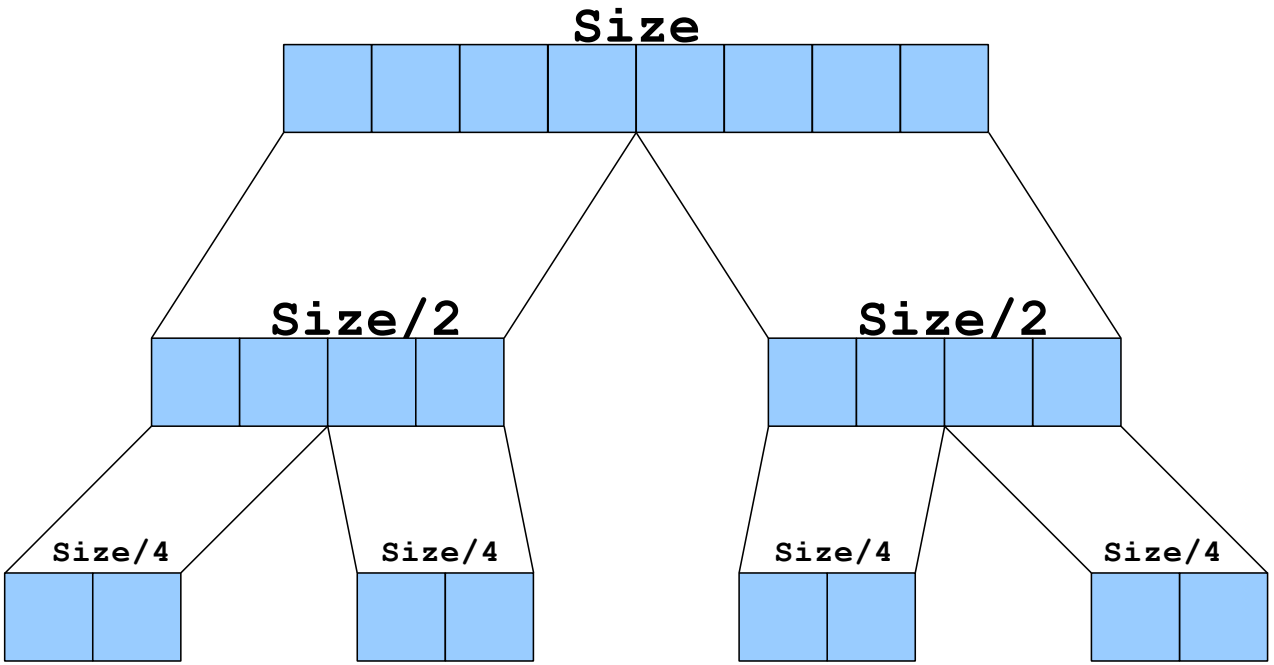
```
1   void combina( int arr[] , int start , int mid , int end )
2   {
3       int iSx = start , iDx = mid; // stato
4       std::vector<int> tempResult; // buffer
5       while(1)
6       {
7           if(arr[iSx] < arr[iDx])
8           {
9               tempResult.push_back(arr[iSx++]);
10              // CONDIZIONE USCITA
11          }
12          else
13          {
14              tempResult.push_back(arr[iDx++]);
15              // CONDIZIONE USCITA
16          }
17      }
18      // GESTISCO ULTIMI
19      // RICOPIO da buffer a arr
    }
```

```
1       2       5       6       8       12      18      26      78

3       6       9       99      100     120     150     168     300
```

```
sx 1
sx 2
dx 3
sx 5
dx 6
sx 6
sx 8
dx 9
sx 12
sx 18
sx 26
sx 78
```

```
1       2       3       5       6       6       8       9     _ 12

18      26      78      99      100     120     150     168     300
```

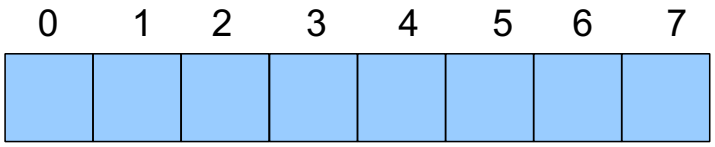
divide



Lista ordinata Triviale

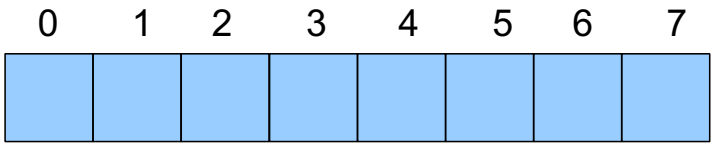
Divide

```
1                ( int * arr , int start , int end )
2   {
3       int mid;
4
5       {
6           mid = (          )   ; // DIVIDE
7
8
9
10      }
11  }
```



Divide

```
1                ( int * arr , int start , int end )
2   {
3       int mid;
4
5       {
6           mid = (start+end)/2; // DIVIDE
7
8
9
10      }
11  }
```



Divide , Conquer

```
1                ( int * arr , int start , int end )
2   {
3       int mid;
4
5       {
6           mid = (start+end)/2; // DIVIDE
7           conquer( arr , start , mid ); // CONQUER
8           conquer( arr , mid+1 , end ); // CONQUER
9
10      }
11  }
```

Divide , Conquer

```
1       conquer ( int * arr , int start , int end )
2   {
3       int mid;
4
5       {
6           mid = (start+end)/2; // DIVIDE
7           conquer( arr , start , mid ); // CONQUER
8           conquer( arr , mid+1 , end ); // CONQUER
9
10      }
11  }
```

Divide , Conquer

```
1       conquer ( int * arr , int start , int end )
2   {
3       int mid;
4       if( start<end )
5       {
6           mid = (start+end)/2; // DIVIDE
7           conquer( arr , start , mid ); // CONQUER
8           conquer( arr , mid+1 , end ); // CONQUER
9
10      }
11  }
12
```

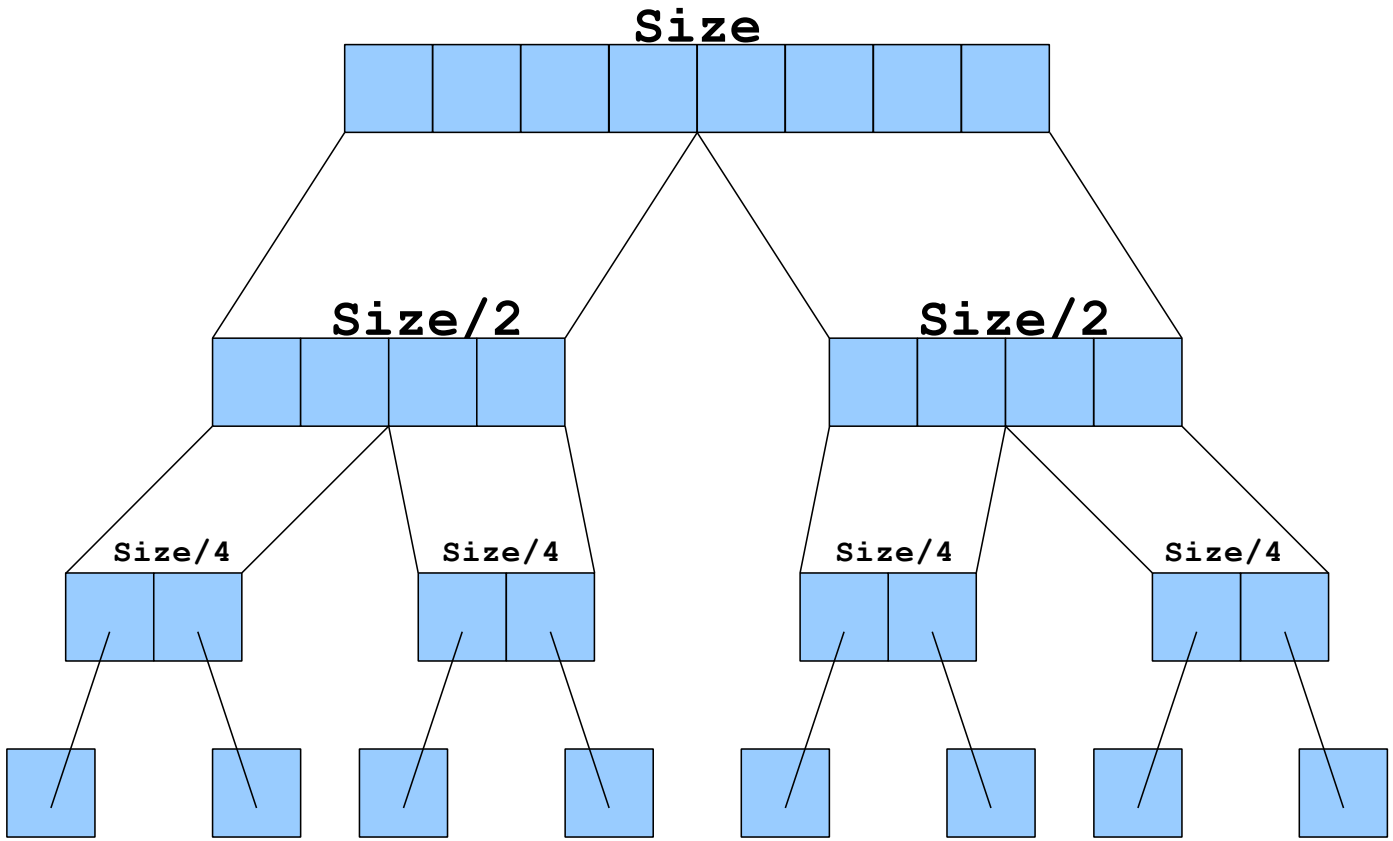
Divide , Conquer , Combine

```
1       conquer ( int * arr , int start , int end )
2   {
3       int mid;
4       if( start<end )
5       {
6           mid = (start+end)/2; // DIVIDE
7           conquer( arr , start , mid ); // CONQUER
8           conquer( arr , mid+1 , end ); // CONQUER
9           combina( arr , start , mid+1 , end );
10      }
11  }
12
```

Divide , Conquer , Combine

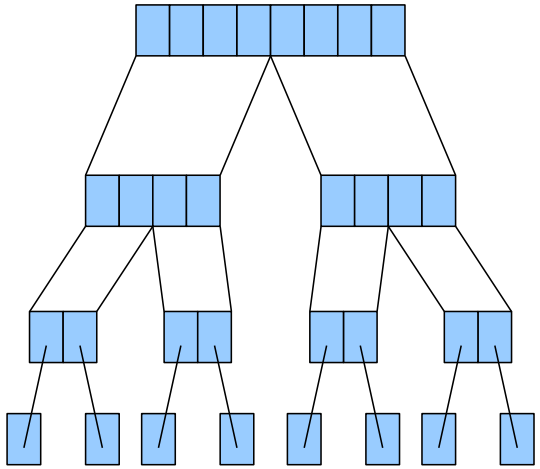
```
1   void mergeSort( int * arr , int start , int end )
2   {
3       int mid;
4       if( start<end )
5       {
6           mid = (start+end)/2; // DIVIDE
7           mergeSort( arr , start , mid ); // CONQUER
8           mergeSort( arr , mid+1 , end ); // CONQUER
9           combina( arr , start , mid+1 , end );
10      }
11  }
```

Ricorsione



Complessità mergesort

- elementi n
- Livelli $log(n)+1$
- Costo livello n



$$n(log(n)+1) \longrightarrow n\,log(n)+n$$

Complessità mergesort

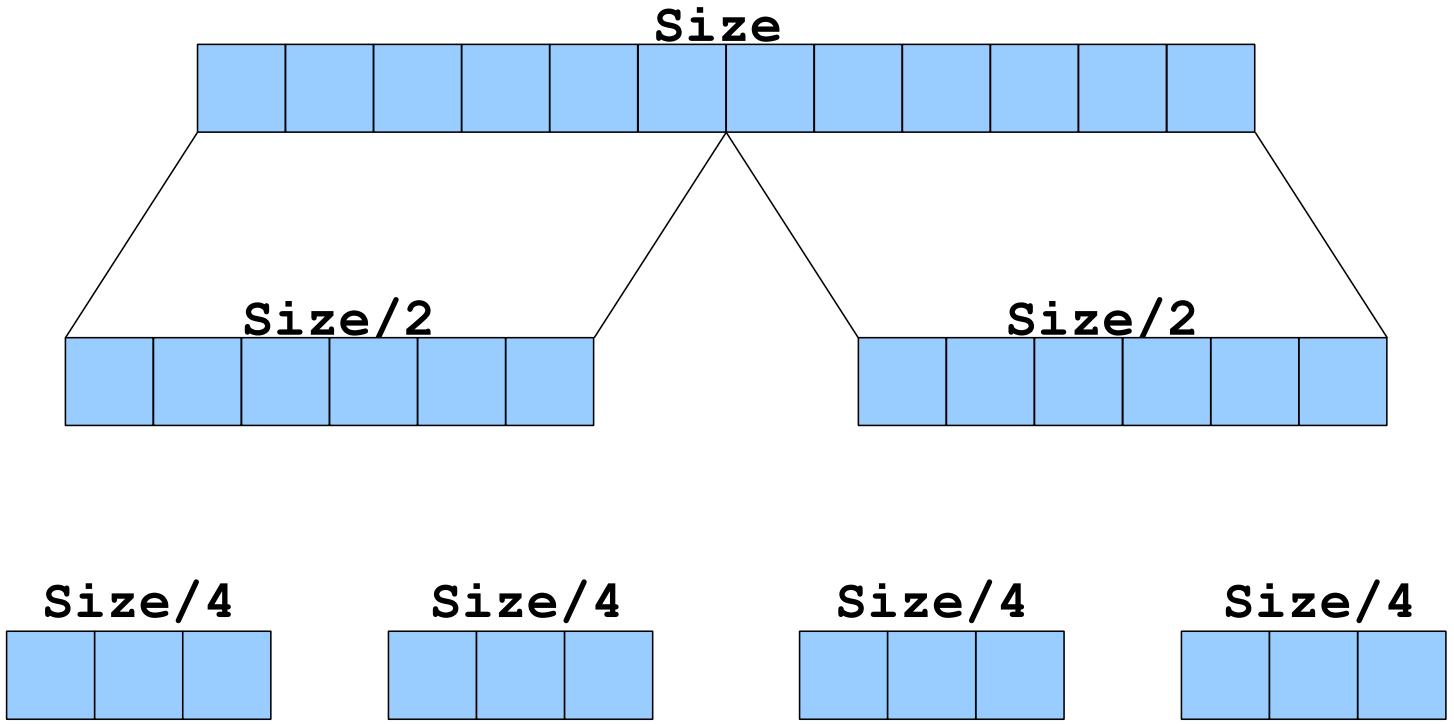
$$\Theta\left(n\,log\left(n\right)\right)$$

BEST CASE

WORST CASE

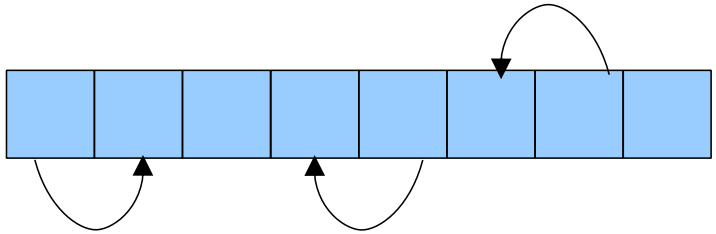
	Worst Case	Best Case	Average Case
Merge Sort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$

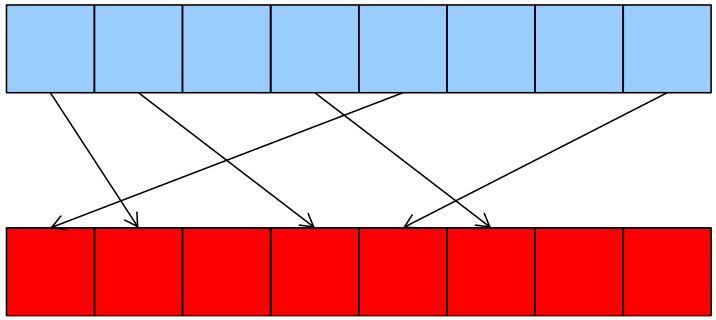
Ibrido



Complessità?

- Tempo di esecuzione: **worst** vs **best** vs **avg**
- Memoria: **in-place** or **not in-place**?

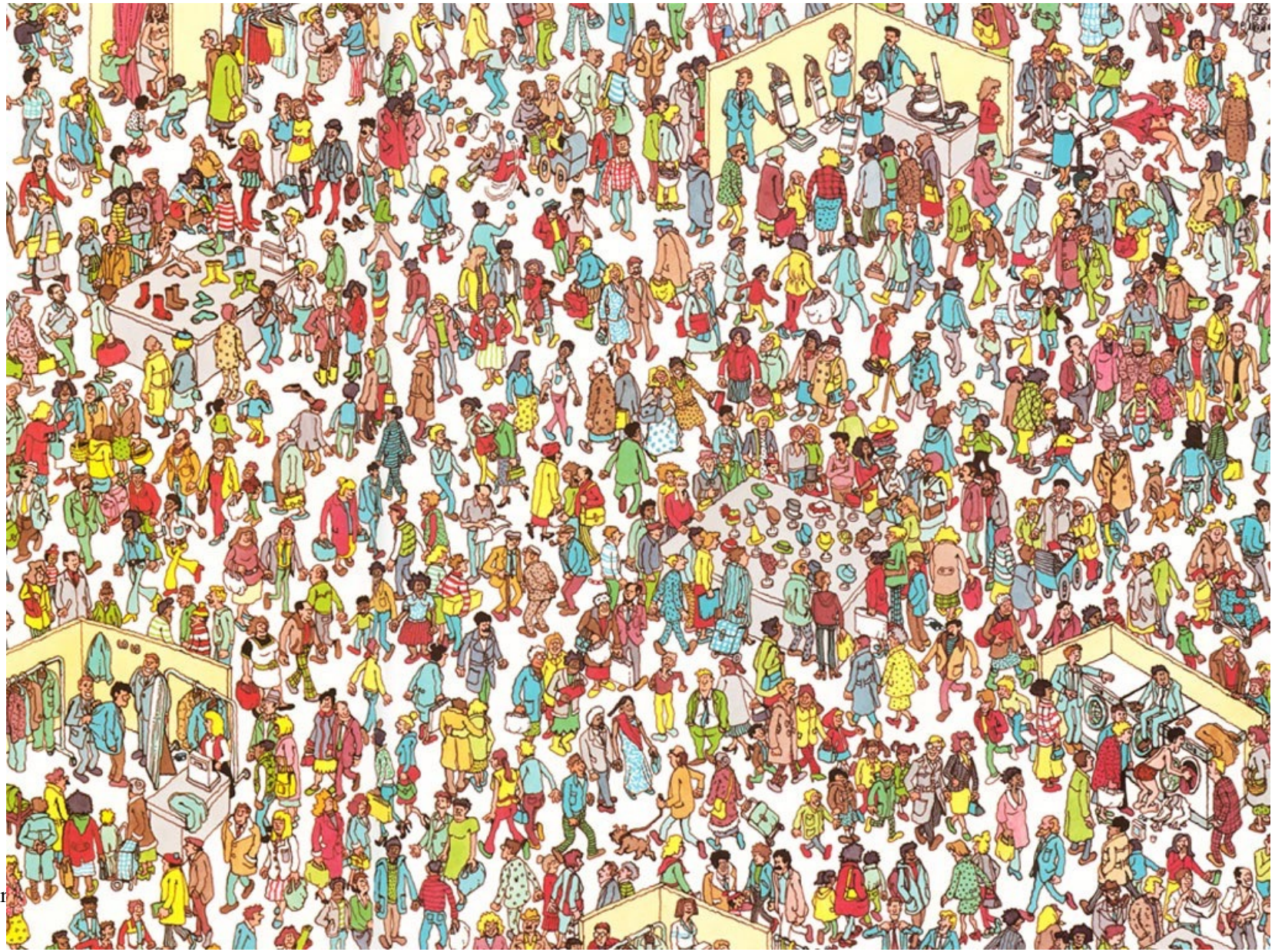


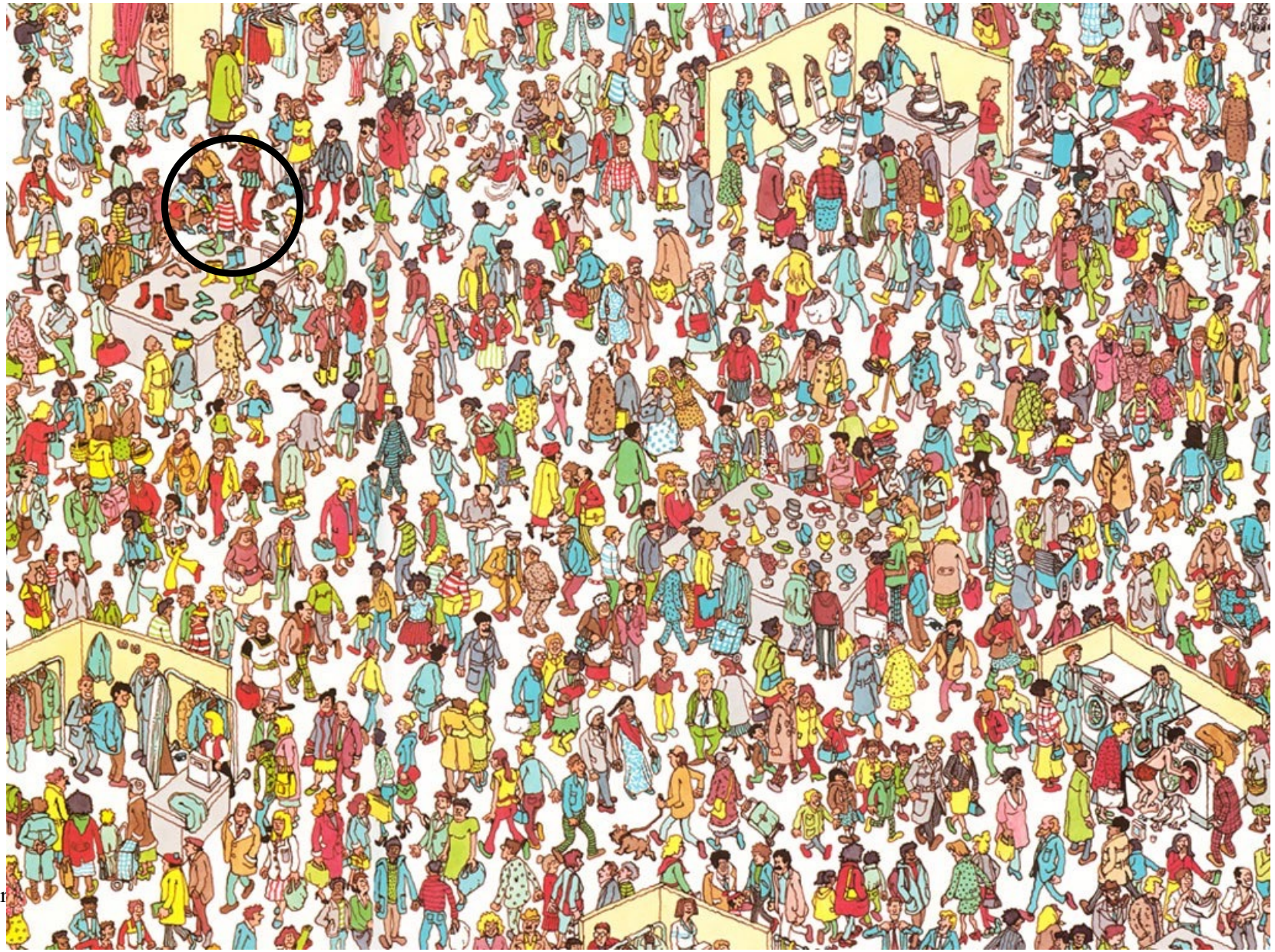


Test

- Insertionsort
- Mergesort
- Casi
  - Random
  - Ordinata
  - Inversa
- Variare quantità linearmente (10 , 40 , 80…)

Where is Wally?





Find Bug-Wally

```
1   [..]
2        int iSx = start , iDx = mid;
3       int stop , iRim;
4       std::vector<int> tempResult;
5       while(1)
6       {if(arr[iSx] < arr[iDx]);
7           {tempResult.push_back(arr[iSx++]);
8               if(iSx == mid)
9               { iRim = iDx;
10                stop = end;
11                break;}
12              continue;
13          }if(arr[iSx] >= arr[iDx])
14          {tempResult.push_back(arr[iDx++]);
15              if(iDx == end+1)
16              { iRim = iSx;
17                stop = mid;
18                break;
19              }
20  [..]
```

```
start
merge sort
33      36      27      15      43      35      36      42      49      21      12      27      40      9       13      26      40
6       11      18      17      29      32      30      12      23      17      35      29      2       22      8       19      17
1       42      29      23      21      19      34      37      48      24      15      20      13      26      41      30      6
0       46      31      5       25      34      27      36      5       46      29      13      7       24      45      32      45
4       14      43      0       37      8       26      28      38      34      3       1       4       49      32      10      26
2       26      36      44      39
```

Compiler Flags

- g++ **-W** -o test test.cpp
- g++ **-Wall** -W -o test test.cpp

```
start
merge sort
33      36      27      15      43      35      36      42      49      21      12      27      40      9       13      26      40
6       11      18      17      29      32      30      12      23      17      35      29      2       22      8       19      17
1       42      29      23      21      19      34      37      48      24      15      20      13      26      41      30      6
0       46      31      5       25      34      27      36      5       46      29      13      7       24      45      32      45
4       14      43      0       37      8       26      28      38      34      3       1       4       49      32      10      26
2       26      36      44      39
kruviser@ilMioComputer:~/Dropbox/lezioni algoritmi/lezione 2$ g++ -W -o testMergeSortBug testMergeSortBug.cpp
testMergeSortBug.cpp: In function ‘void combina(int*, int, int, int)’:
testMergeSortBug.cpp:135:32: warning: suggest braces around empty body in an ‘if’ statement [-Wempty-body]
```

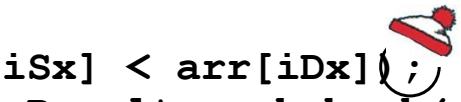
Warning: suggest braces around empty body in an ‘if’ statement

Find Bug-Wally

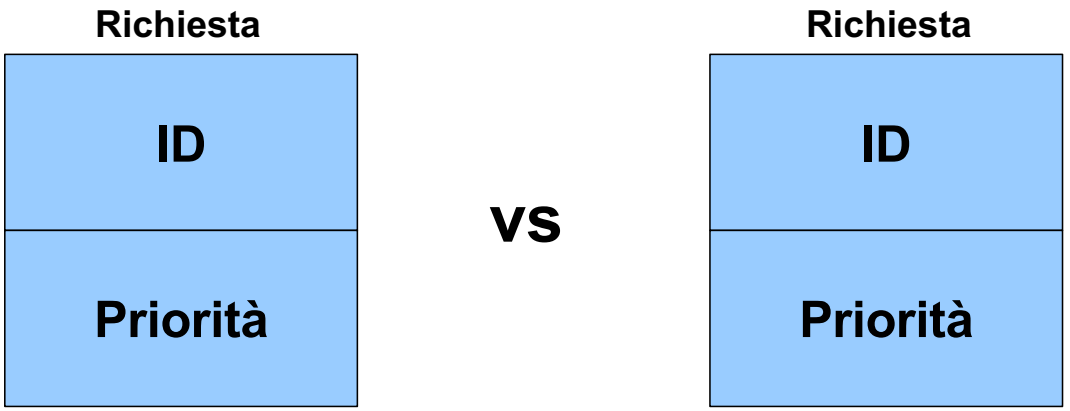
```
1   [..]
2        int iSx = start , iDx = mid;
3       int stop , iRim;
4       std::vector<int> tempResult;
5       while(1)
6       {if(arr[iSx] < arr[iDx]);
7           {tempResult.push_back(arr[iSx++]);
8               if(iSx == mid)
9               { iRim = iDx;
10                stop = end;
11                break;}
12              continue;
13          }if(arr[iSx] >= arr[iDx])
14          {tempResult.push_back(arr[iDx++]);
15              if(iDx == end+1)
16              { iRim = iSx;
17                stop = mid;
18                break;
19              }
20  [..]
```

Find Bug-Waldo

```
1   [..]
2        int iSx = start , iDx = mid;
3       int stop , iRim;
4       std::vector<int> tempResult;
5       while(1)
6       {if(arr[iSx] < arr[iDx]);
7           {tempResult.push_back(arr[iSx++]);
8               if(iSx == mid)
9               { iRim = iDx;
10                stop = end;
11                break;}
12              continue;
13          }if(arr[iSx] >= arr[iDx])
14          {tempResult.push_back(arr[iDx++]);
15              if(iDx == end+1)
16              { iRim = iSx;
17                stop = mid;
18                break;
19              }
20  [..]
```



Ordinamenti multi-valore



- Richieste servite in ordine di ID crescente
- A parità di ID, si serve in ordine di priorità decrescente

Richiesta

```
1   struct Richiesta
2   {
3       int id_;
4       int prio_;
5
6       // costruttore con lista init
7       Richiesta(int id, int prio):
8                 id_(id) , prio_(prio){}
9   };
10
11
```

STL: sort()

```
sort ( first, last );
```

STL: sort()

```
sort ( first, last, comparatore );
```

Estremi del vettore da ordinare

Funzione di confronto
- True
- False

Richiesta

```
1   bool confrontaRichieste( Richiesta r1 , Richiesta r2)
2   {
3       // SE ID1 < ID2
4          // VINCE 1
5
6
7
8
9
10
11
12
13
14
15  }
```

Richiesta

```
1   bool confrontaRichieste( Richiesta r1 , Richiesta r2)
2   {
3       // SE ID1 < ID2
4           // VINCE 1
5
6       // SE ID1 == ID2
7       {
8
9
10
11
12      }
13
14
15  }
```

Richiesta

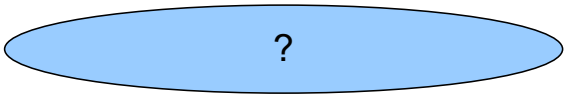
```
1   bool confrontaRichieste( Richiesta r1 , Richiesta r2)
2   {
3       // SE ID1 < ID2
4           // VINCE 1
5
6       // SE ID1 == ID2
7       {
8           // SE PRIO1 > PRIO2
9               // VINCE 1
10
11
12      }
13
14
15  }
```

Richiesta

```
1   bool confrontaRichieste( Richiesta r1 , Richiesta r2)
2   {
3       // SE ID1 < ID2
4           // VINCE 1
5
6       // SE ID1 == ID2
7       {
8           // SE PRIO1 > PRIO2
9               // VINCE 1
10
11
12      }
13      // TUTTI GLI ALTRI CASI
14          // VINCE 2
15  }
```

Richiesta

```
1   bool confrontaRichieste( Richiesta r1 , Richiesta r2)
2   {
3       if( r1.id_<r2.id_ )
4           return true;
5
6       else if(r1.id_ == r2.id_)
7       {
8           if(r1.prio_>r2.prio_)
9               return true;
10
11
12      }
13      else
14          return false;
15  }
```



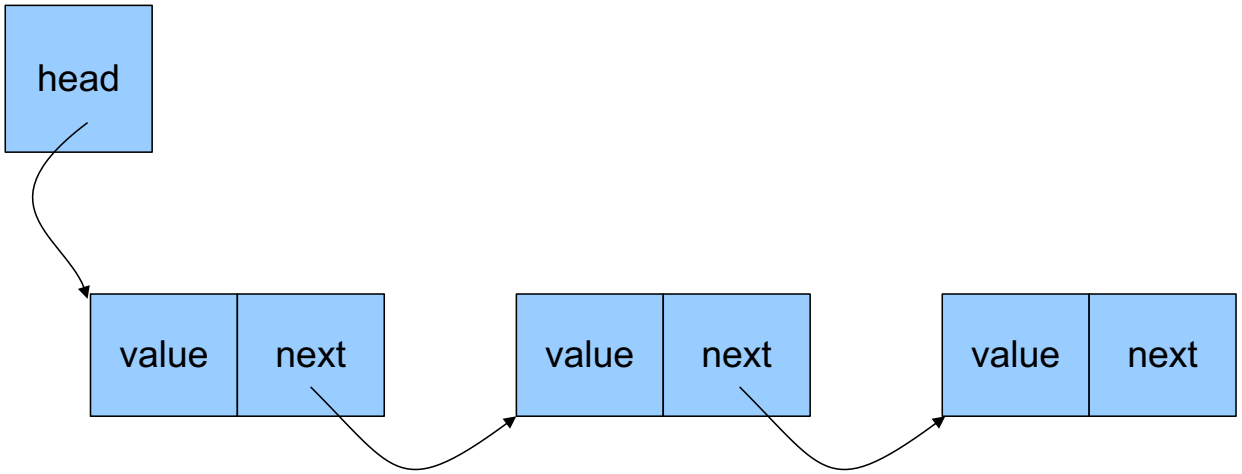
Richiesta

```
1   bool confrontaRichieste( Richiesta r1 , Richiesta r2)
2   {
3       if( r1.id_<r2.id_ )
4           return true;
5
6       else if(r1.id_ == r2.id_)
7       {
8           if(r1.prio_>r2.prio_)
9               return true;
10          else
11              return false;
12      }
13      else
14          return false;
15  }
```

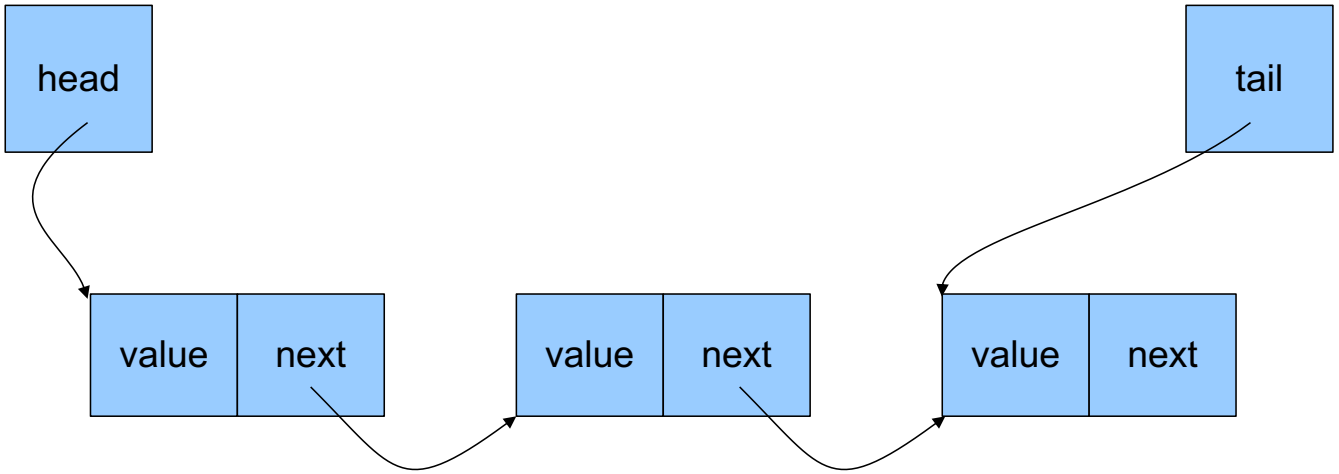
Tipo Accessi

vs

liste



liste



Solo inserimento in coda

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       // LEGGO LUNGHEZZA
4
5
6       // VARIABILI DI APPOGGIO
7
8       // PER TUTTA LA LUNGHEZZA
9       {
10          // LEGGO VALORE
11
12          // CREO E INIZIALIZZO OGGETTO
13
14
15          // AGGIORNO TESTA
16      }
17      // RITORNO TESTA
18  }
```

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , newObj;
7
8
9
10
11
12
13
14
15
16
17  }
18
```

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , newObj;
7
8       for( int i = 0 ; i < l ; ++i )
9       {
10
11
12
13
14
15
16      }
17
18  }
```

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , newObj;
7
8       for( int i = 0 ; i < l ; ++i )
9       {
10          cin >> value;
11          newObj = new Obj();
12          newObj->next_ = head;
13          newObj->value_ = value;
14
15          head = newObj;
16      }
17
18  }
```

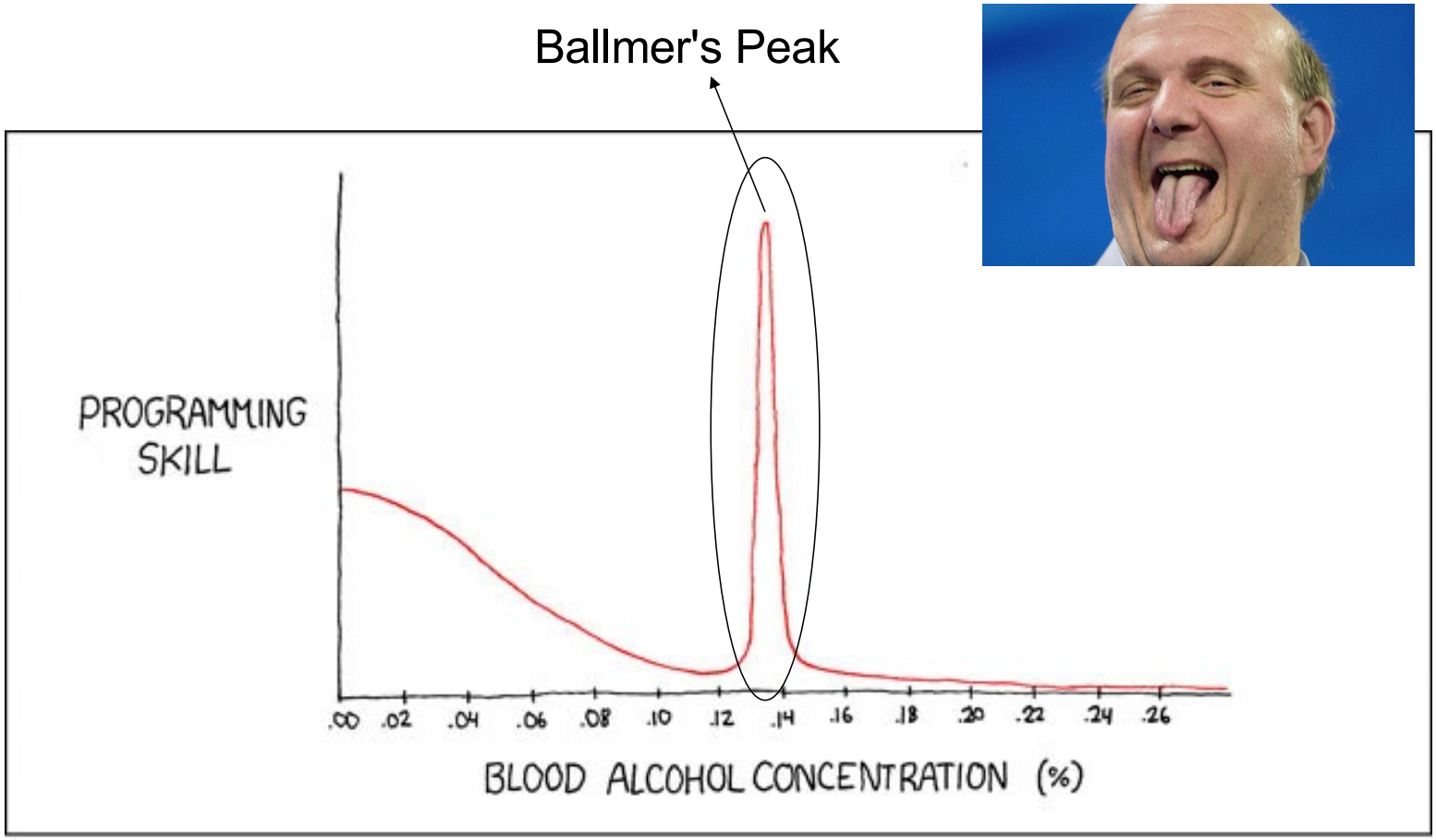
Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , newObj;
7
8       for( int i = 0 ; i < l ; ++i )
9       {
10          cin >> value;
11          newObj = new Obj();
12          newObj->next_ = head;
13          newObj->value_ = value;
14
15          head = newObj;
16      }
17      return head;
18  }
```

Stampa Lista

```
1   void stampaLista( Obj * head )
2   {
3       Obj * pointer = head;
4       while( pointer != NULL )
5       {
6           cout << pointer->value_ << endl ;
7           pointer = pointer->next_;
8       }
9       cout << endl;
10  }
11
12
```

Birra!



source: https://xkcd.com/323/

```
Hello world

Segmentation fault
```

```
kruviser@ilMioComputer:~/Dropbox/lezioni algoritmi/lezione 2$ ./testList < listFile
letto 10
5       1       26      8       12      78      6       2       18      3
3
18
2
6
78
12
8
26
1
5
Segmentation fault (core dumped)
```

Valgrind

- Babysitter Memoria

- Controlla accessi
- Conta accessi

valgrind ./eseguibile

```
12
8
26
1
5
==3307== Conditional jump or move depends on uninitialised value(s)
==3307==    at 0x8048719: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==
==3307== Use of uninitialised value of size 4
==3307==    at 0x80486E5: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==
==3307== Invalid read of size 4
==3307==    at 0x80486E5: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==  Address 0xffff is not stack'd, malloc'd or (recently) free'd
==3307==
==3307==
==3307== Process terminating with default action of signal 11 (SIGSEGV)
==3307==  Access not within mapped region at address 0xFFFF
==3307==    at 0x80486E5: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==  If you believe this happened as a result of a stack
==3307==  overflow in your program's main thread (unlikely but
==3307==  possible), you can try to increase the size of the
==3307==  main thread stack using the --main-stacksize= flag.
==3307==  The main thread stack size used in this run was 8388608.
==3307==
==3307== HEAP SUMMARY:
==3307==     in use at exit: 80 bytes in 10 blocks
==3307==   total heap usage: 10 allocs, 0 frees, 80 bytes allocated
==3307==
==3307== LEAK SUMMARY:
==3307==    definitely lost: 0 bytes in 0 blocks
==3307==    indirectly lost: 0 bytes in 0 blocks
==3307==      possibly lost: 0 bytes in 0 blocks
==3307==    still reachable: 80 bytes in 10 blocks
```

```
g++ -g -o eseguibile eseguibile.cpp
```

```
valgrind ./eseguibile
```

```
12
8
26
1
5
==3288== Conditional jump or move depends on uninitialised value(s)
==3288==    at 0x8048719: stampaLista(Obj*) (testList.cpp:21)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==
==3288== Use of uninitialised value of size 4
==3288==    at 0x80486E5: stampaLista(Obj*) (testList.cpp:23)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==
==3288== Invalid read of size 4
==3288==    at 0x80486E5: stampaLista(Obj*) (testList.cpp:23)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==  Address 0xffff is not stack'd, malloc'd or (recently) free'd
==3288==
==3288==
==3288== Process terminating with default action of signal 11 (SIGSEGV)
==3288==  Access not within mapped region at address 0xFFFF
==3288==    at 0x80486E5: stampaLista(Obj*) (testList.cpp:23)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==  If you believe this happened as a result of a stack
==3288==  overflow in your program's main thread (unlikely but
==3288==  possible), you can try to increase the size of the
==3288==  main thread stack using the --main-stacksize= flag.
==3288==  The main thread stack size used in this run was 8388608.
==3288==
==3288== HEAP SUMMARY:
==3288==     in use at exit: 80 bytes in 10 blocks
==3288==   total heap usage: 10 allocs, 0 frees, 80 bytes allocated
==3288==
==3288== LEAK SUMMARY:
==3288==    definitely lost: 0 bytes in 0 blocks
==3288==    indirectly lost: 0 bytes in 0 blocks
==3288==      possibly lost: 0 bytes in 0 blocks
==3288==    still reachable: 80 bytes in 10 blocks
```

Stampa Lista

```
1   void stampaLista( Obj * head )
2   {
3       Obj * pointer = head;
4       while( pointer != NULL )
5       {
6           cout << pointer->value_ << endl ;
7           pointer = pointer->next_;
8       }
9       cout << endl;
10  }
11
12
```

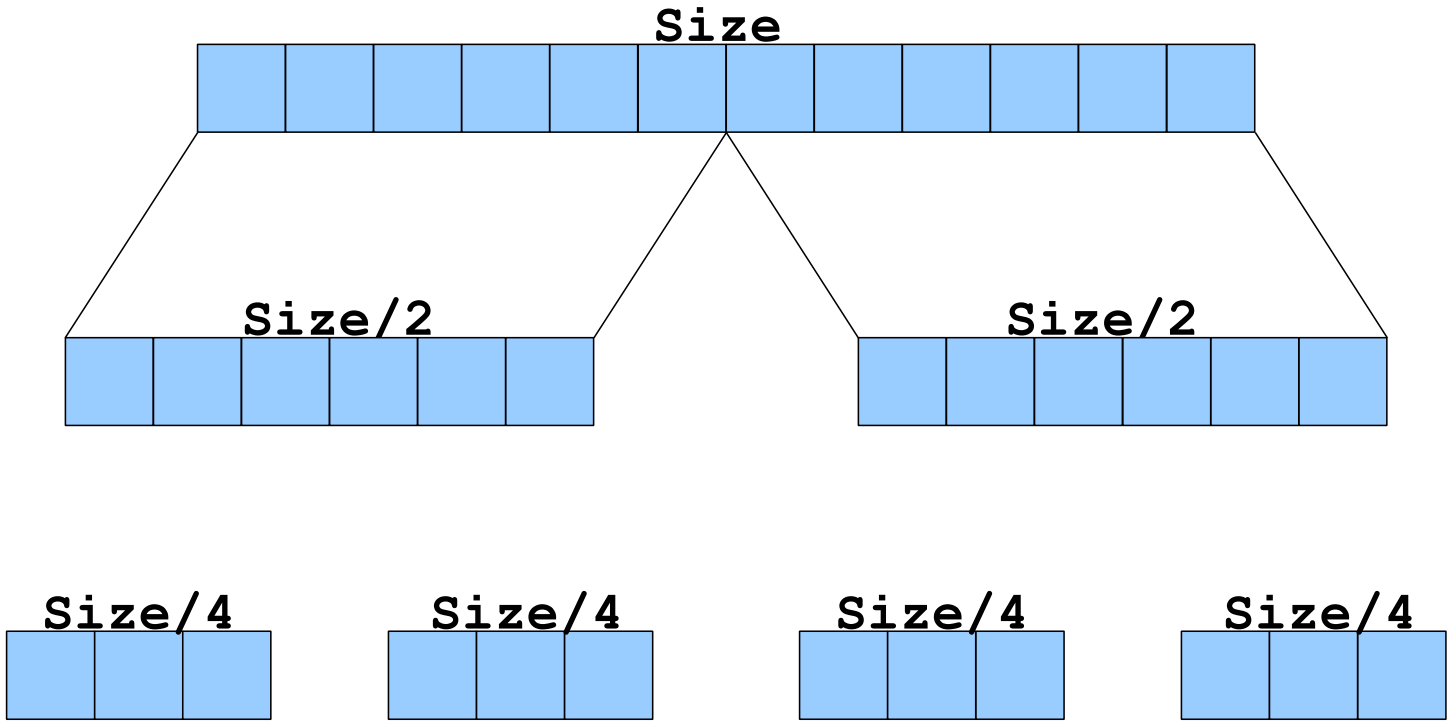
Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , newObj;
7
8       for( int i = 0 ; i < l ; ++i )
9       {
10          cin >> value;
11          newObj = new Obj();
12          newObj->next_ = head;
13          newObj->value_ = value;
14
15          head = newObj;
16      }
17      return head;
18  }
```

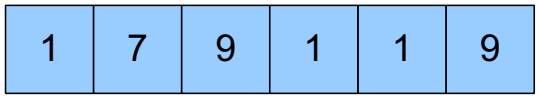
Operazioni su Lista

- Ricerco un elemento e lo sposto in testa
  - Scorrere
  - Estrazione
  - Inserzione testa
- Ricerco un elemento e lo sposto in coda
  - Scorrere
  - Estrazione
  - Inserimento in coda...

Merge Sort Ibrido

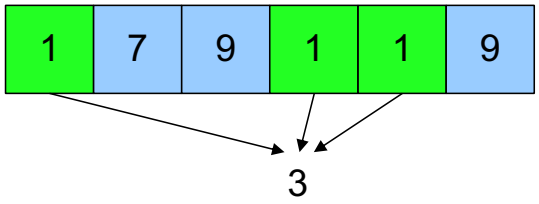


Distinti in Array



- Input: elementi array
- Output: array senza duplicati

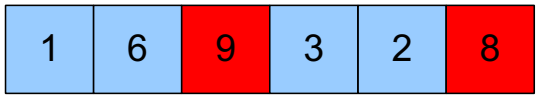
K interi più frequenti



- Input: elementi array , intero k
- Output: primi k valori più frequenti

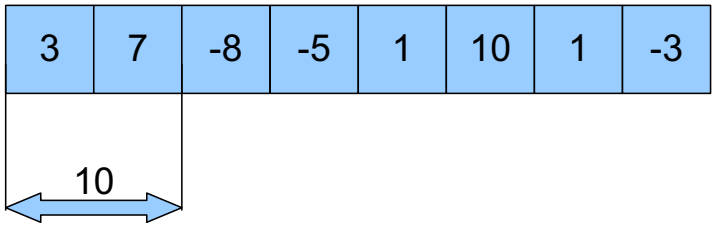
K interi più grandi

K=2

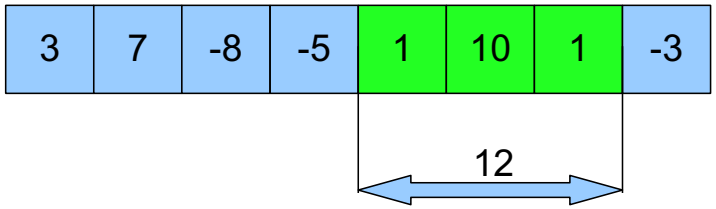


- Input: elementi array , intero k
- Output: primi k valori ordinati in maniera decrescente

Somma Massima



- Input: array
- Output: somma massima di gruppi di elementi adiacenti
- Esempio



Soluzione 1

```
1   int somme1(int a[] , int size )
2   {
3
4
5
6       for(i=0; i<size; i++)                    // n
7       {
8
9
10
11
12
13
14
15
16
17      }
18      return max;
    }
```

Soluzione 1

```
1   int somme1(int a[] , int size )
2   {
3
4
5
6       for(i=0; i<size; i++)                    // n
7       {
8           for(j=i; j<size; j++)                // n
9           {
10
11
12
13
14
15
16          }
17      }
18      return max;
    }
```

Soluzione 1

```
1   int somme1(int a[] , int size )
2   {
3       int somma;
4       int i,j,k;
5       int max=a[0];
6       for(i=0; i<size; i++)                    // n
7       {
8           for(j=i; j<size; j++)                // n
9           {
10              somma=0;
11              for(k=i; k<=j; k++)              // n
12              {
13                  somma+=a[k];
14              }
15              if(somma > max) max=somma;
16          }
17      }
18      return max;
    }
```

Soluzione 1

```
1   int somme1(int a[] , int size )
2   {
3       int somma;
4       int i,j,k;
5       int max=a[0];
6       for(i=0; i<size; i++)                    // n
7       {
8           for(j=i; j<size; j++)                // n
9           {
10              somma=0;
11              for(k=i; k<=j; k++)              // n
12              {
13                  somma+=a[k];
14              }
15              if(somma > max) max=somma;
16          }
17      }
18      return max;
    }
```

$$\Theta\left(n^3\right)$$

Soluzione 2

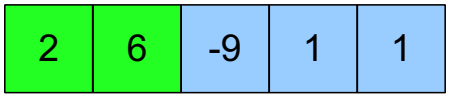
```
1   int somme2(int a[] , int size )
2   {
3       int somma;
4       int i,j;
5       int max=a[0];
6       for(i=0; i<size; i++)
7       {
8           somma=0;
9           for(j=i; j<size; j++)
10          {
11              somma+=a[j];
12              if(somma > max) max=somma;
13          }
14      }
15      return max;
16  }
17
18
```

Soluzione 2

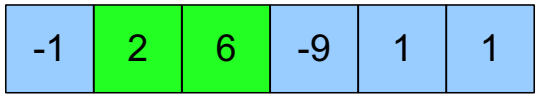
```
1   int somme2(int a[] , int size )
2   {
3       int somma;
4       int i,j;
5       int max=a[0];
6       for(i=0; i<size; i++)                    // n
7       {
8           somma=0;
9           for(j=i; j<size; j++)                // n
10          {
11              somma+=a[j];
12              if(somma > max) max=somma;
13          }
14      }
15      return max;                              Θ(n^2)
16  }
17
18
```

proprietà

la somma degli elementi del sotto array di somma massima è sempre positiva



Il valore precedente al primo valore del sotto array di somma massima è negativo



Esercizi

- Esperimenti
  - Merge vs Insertion sort vs Ibrido
  - Soluzioni array somma massima
  - Input critici (array inverso, array ordinato)
- Esercizi
  - Inserimenti testa/coda liste
  - Distinti
  - Più frequenti
  - Più grandi