Gruppo **Informathink**

Individuato un argomento a scelta, formulare le domande per le 12 tipologie viste, indicando per ognuno l'obiettivo formativo che si vuole raggiungere/testare.

Argomento: **Gli array.**

Tipo 1. **Sviluppare una soluzione**

Domanda:

Scrivere o spiegare un algoritmo capace di ordinare un array al più in $O(n^2)$ iterazioni.

Risposta:

```
Public void sort(ArrayList<int> a){
   int i = 0;
   int s = a.size;
   int j=0;
    while (i < s - 1){
        j++;
        while (j <= s){
            if (a.get(i) > a.get(j))
                swap(a.get(i), a.get(j));
        }
     }
}
```

Obiettivo formativo:

Un array può essere ordinato in un elevato numero di modi, e l'unico limite che si impone è quello della complessità computazionale. La domanda è posta in modo tale da far riflettere ogni singolo alunno sull'approccio algoritmico da utilizzare. Alcuni avrebbero potuto utilizzare un solo indice, altri invece, come nella soluzione proposta nella risposta, due indici. Altri ancora avrebbero potuto utilizzare un algoritmo ricorsivo. Non avendo inserito alcun vincolo sull'espressione dell'algoritmo, l'alunno non è limitato ad un linguaggio e ai suoi relativi costrutti, oppure alla programmazione in generale. Gli è solo richiesta la comprensione dell'array come struttura dati.

Tipo 2. **Sviluppare una soluzione che usa un particolare modulo**

Domanda:

Scrivere o spiegare un algoritmo capace di trovare il minimo in un array di interi, usando la funzione `max(x, y)`, definita come segue:

```
Public int max(int x, int y){
    if (x > y)
        return x;
    return y;
}
```

Risposta:

```
public int find_min(ArrayList<int> a){
    int min = 8967858578; //assegno un numero molto grande
    for(int i=0;i<=a.size; i++){
              if(max(min,a.get(i))== min)
                   min=a.get(i);
     }
    return min;
}
```

Obiettivo formativo:

Insegnare agli alunni che non sempre si può scegliere tutto ciò che riguarda l'implementazione di una soluzione ad un dato problema. Spesso ci si può trovare a lavorare con strumenti che non sono relativi in modo specifico al problema da trattare, ma che comunque, utilizzando un po' di "furbizia" e logica, si prestano comunque al raggiungimento della soluzione.

Tipo 3. **Tracciare una data soluzione**

Domanda:

Cosa succede alla funzione se chiamata con parametro `a = [0, 1, 2]`, e come varia il parametro `a` nelle varie chiamate a funzione di `rec_print`?

```
public int rec_print(ArrayList<int>a){
    int c=0;
    if (a.size == 0)
        return -1;
    c = a.get(0);
    System.out.println(c);
    a.remove (c);
    rec_print(a);
     return c;
}
```

Risposta:

La funzione `rec_print([0, 1, 2])` stampa il numero 0, per poi rimuovere lo 0 dalla lista e invocarsi ricorsivamente, questa volta sull'input `[1, 2]`.

Viene poi eseguita la stampa del numero 1, che viene rimosso e si invoca ricorsivamente la funzione con parametro `[2]`.

Di nuovo, viene stampato il numero 2, viene rimosso e si invoca ricorsivamente la funzione con parametro `[]`.

Qui il controllo restituisce true in quanto la dimensione dell'array è 0, e quindi la funzione ricorsiva termina. Così anche le precedenti terminano con essa, uscendo dalla chiamata ricorsiva ed eseguendo l'istruzione return.

Obiettivo formativo:

Si obbliga lo studente ad esaminare il codice presentato su un dato input. Serve per far sviluppare intuito sulle funzioni ricorsive, e c'è una sola risposta corretta. Anche comprendendo le singole istruzioni, lo studente deve obbligatoriamente ragionare ad uno stato più astratto sull'intera funzione e il flusso di esecuzione del codice.

Tipo 4. **Analisi di un codice**

Domanda:

Quante iterazioni esegue questo codice?

```
int a = [0];
i = 1;
do{
    a.add(i);
    i = i + 1;
}while(a.lenght != 0);
```

Risposta:

Il codice esegue infinite iterazioni, in quanto viene effettuato un ciclo su ogni valore presente nell'array, ma ad ogni iterazione del ciclo viene aggiunto un valore all'array.

Obiettivo formativo:

Per rispondere a questa domanda, è richiesto che lo studente abbia comprensione di come funziona la struttura dati che si sta adoperando. Bisogna analizzare l'esecuzione del codice e comprendere tutti i cambiamenti alla struttura stessa, insieme alla logica – o meglio, errore logico – che è stato previsto.

Tipo 5. **Trovare il problema risolto da una data soluzione**

Domanda:

Cosa effettua il seguente frammento di codice?

```
public void mistery (ArrayList<int> a){
 int   i = 0;
 int s = a.size;
int j=0;
    while (i < s - 1){
       j = i + 1;
        while (j <= s){
            if (a.get(i) < a.get(j)){
                tmp = a.get(i);
                a.get(i) = a.get(j);
                a.get(j) = tmp;
          }
     }
}
```

Risposta:

Ordina l'array `a` in ordine decrescente.

Obiettivo formativo:

Agli studenti viene chiesto non solamente di comprendere il flusso di esecuzione del codice, ma anche il ragionamento che c'è stato dietro la scrittura del codice stesso. Vengono infatti messi davanti alla situazione di dover gestire del codice non documentato, o comunque pensato da una persona esterna che potrebbe non avere il loro stesso approccio algoritmico.

Tipo 6. **Esaminare la correttezza di una data soluzione**

Domanda:

Il seguente frammento di codice è stato scritto nella scorsa prova. La traccia era quella di scrivere una funzione che restituisse true se tutti gli elementi di un array sono pari, false altrimenti.

```
public boolean even_or_odd(ArrayList<int> a){
    for (int i=0;i<=a.size;i++){
        if(a.get(i)% 2 != 0)
            return false;
    }
    return true;
}
```

È corretto?

Risposta:

Sì. L'array viene scansionato, e per ogni suo valore viene controllato il resto della divisione tra il valore stesso e il numero 2. Nel caso in cui il resto sia diverso da 0, vuol dire che il numero non è divisibile per due, e quindi, che è dispari. In questo caso siamo di fronte all'evenienza in cui c'è almeno un valore dispari, e quindi che l'array non può contenere solo elementi pari. Mi è sufficiente questa scoperta per restituire false. In alternativa, una volta che ho processato tutti gli elementi, se sono uscito dal for significa che non ho trovato elementi dispari, e che quindi ci sono solo elementi pari nell'array, e posso restituire true.

Obiettivo formativo:

Anche qui, come nella domanda di tipo 5, uno studente analizza una soluzione che potrebbe divergere dal modo in cui la implementerebbe, e viene addestrata la sua flessibilità in materia.

Tipo 7. **Completare una data soluzione**

Domanda:

Completa il seguente frammento di codice in modo tale che, dato un array `a` di interi (anche negativi), io riesca a trovare il massimo

```
public int find_max(ArrayList<int> a){
  unsigned max = _;
    _ # Riga opzionale da usare, si può lasciare vuota
    for (int i=0;i<=a.size;i++){
          if _
            max =a.get(i);
     }

    return max
}
```

Risposta:

```
public int find_max(ArrayList<int> a){
  unsigned max = a.get(0);
    _ # Riga opzionale da usare, si può lasciare vuota
    for (int i=0;i<=a.size;i++){
          if(max<a.get(i))
            max =a.get(i);
     }
    return max
}
```

Obiettivo formativo:

La domanda può essere risolta in vari modi. Una soluzione alternativa è quella di impostare il valore di max al primo valore dell'array, in modo da ipotizzare che se ci fosse un solo valore nell'array esso sarà sicuramente il più grande.Viene chiesta questa domanda per capire se lo studente ha compreso il concetto di "massimo di un array", e ha compreso come manipolare (o non) la struttura per trovarlo.

Tipo 8. **Manipolazione istruzioni**

Domanda:

Cosa succede se nel seguente codice, che esegue l'ordinamento di un array,

```
1. public void sort(ArrayList<int> a):
2.     int i = 0; int j=0;
3.     int s = a.size;
4.      while (i < s - 1){
5.          j = i + 1;
6.          while (j <= s){
7.              if (a.get(i) >= a.get(j))
8.                  swap(a.get(i), a.get(j));}
9.      }
```

a) si cambia alla riga 7 il >= in un >

b) si cambia alla riga 7 il >= in un <=

Risposta:

a) Si evita di scambiare gli elementi anche se sono uguali, il che è un'operazione inutile
b) Si ordina l'array in ordine decrescente piuttosto che in ordine crescente.

Obiettivo formativo:

Si vuole che lo studente capisca che le sequenze di istruzioni sono generalizzabili, nel senso che è sufficiente cambiare una singola istruzione per risolvere un problema diverso. In questo caso, è sufficiente cambiare la condizione per evitare operazioni inutili, e/o invertire l'ordine di ordinamento dell'array.

Tipo 9. **Stima dell'efficienza**

Domanda:

Qual è la complessità computazionale temporale, espressa in termini di O grande, del seguente frammento di codice?

```
1. public void sort(ArrayList<int> a):
2.     int i = 0; int j=0;
3.     int s = a.size;
4.      while (i < s - 1){
5.          j = i + 1;
6.          while (j <= s){
7.              if (a.get(i) < a.get(j))
8.                  swap(a.get(i), a.get(j));}
9.      }
```

Risposta:

Si parte da 0 e si arriva fino alla dimensione dell'array. Il ciclo while interno (riga 6) viene eseguito un numero di volte pari alla differenza tra la dimensione dell'array ed i. Il ciclo while esterno (riga 4), viene eseguito un numero di volte pari alla dimensione dell'array.

Ad una k-esima iterazione, i vale k e j vale $n - k$ e vengono effettuate k operazioni. All'inizio k vale 1, per cui $n - k$ vale $n - 1$. Poi per $k = 2$, $n - k$ vale $n - 2$. Insomma, vengono effettuate $(n - 1) + (n - 2) + \cdots + (n - (n - 1)) + (n - (n - 0))$ operazioni. Questo può essere scritto come: $1 + 2 + \cdots + (n - 2) + (n - 1)$, che è asintoticamente corrispondente alla somma dei primi n numeri, che sappiamo valere $\frac{n(n+1)}{2}$, ossia $\frac{n^2}{2} + \frac{n}{2}$. Ai fini della complessità computazionale, si considera solo n^2. L'algoritmo risulta essere $O(n^2)$

Obiettivo formativo:

L'efficienza di spazio e di tempo è una valutazione fondamentale per le prestazioni di un algoritmo. Occorre far capire allo studente che questo tipo di valutazioni deve essere effettuato su ogni algoritmo, e non solo su quelli complicati.

Tipo 10. **Strutturazione di una domanda**

Domanda:

Scrivi una domanda riguardo l'ordinamento degli elementi di un array, a livello di complessità di tempo.

Risposta:

Il numero di controlli fatti dal bubble-sort nell'array [1, 2, 3, 4] e [4, 3, 2, 1] è lo stesso?

Obiettivo formativo:

Questo tipo di domande invita i discenti ad adottare un punto di vista diverso. Oltre all'esperienza di risposta, essi sono invitati a realizzare una domanda dal punto di vista dell'educatore.
Questo tipo di domanda li incoraggia a riflettere su quanto hanno studiato e a provare a fare delle domande coerenti con l'argomento e non solo a rispondere.
Inoltre, il design delle domande è una sorta di apprendimento attivo che incoraggia la creatività.

Tipo 11. **Domande sullo stile di programmazione**

Domanda:

Considera queste due soluzioni per trovare il massimo in un array:

(a)

```
public unsigned find_max(ArrayList<int> a){
   unsigned max = a.get(0);
    for (int i=0;i<=a.size;i++){
        if (max < a.get(i))
                 max =a.get(i);
     }
    return max;
}
```

(b)

```
public unsigned find_max(ArrayList<int> a):
   unsigned max = -8765434567890;
    for (int i=0;i<=a.size;i++){
        if (max < a.get(i))
              max =a.get(i);
     }
    return max;
}
```

Quale è la migliore?

Risposta:

Dipende. Se l'array non deve essere modificato durante la ricerca del massimo, allora la seconda è l'unica soluzione accettabile. Nel caso in cui siano permesse modifiche, o l'array utilizzato è una copia di quello originale, allora anche la prima è utilizzabile. La seconda soluzione risulta fare un'operazione in più rispetto alla prima soluzione: mentre la prima confronta direttamente il primo valore dell'array con il secondo, la seconda soluzione confronta un valore molto piccolo con il primo valore, soluzione non sempre valida perché l'array potrebbe contenere valori ancora più piccoli.

Obiettivo formativo:

Dimostrare allo studente che, nonostante il fatto che per ogni problema siano disponibili infinite soluzioni dal punto di vista della programmazione, queste possono differire. L'obiettivo è porre l'attenzione su queste differenze.

Tipo 12. Trasformazione di una soluzione

Domanda:

Implementa il seguente codice in C utilizzando solo cicli for al posto dei while.

```
1. public void sort(ArrayList<int> a):
2.     int i = 0; int j=0;
3.     int s = a.size;
4.      while (i < s - 1){
5.          j = i + 1;
6.          while (j <= s){
7.              if (a.get(i) < a.get(j))
8.                  swap(a.get(i), a.get(j));}
9.      }
```

Risposta:

```
void sort(int* a, int size) {
    for(int i = 0; i < size - 1; i++) {
        for(int j = i+1; j < s; j++) {
            if(a[i] < a[j]) {
                tmp = a[i];
                a[i] = a[j];
                a[j] = tmp;
            }
        }
    }
}
```

Obiettivo formativo:

Il focus di questo tipo di domande si concentra su aspetti della sintassi piuttosto che di logica. In questo caso all'analisi del problema riguarda la trasformazione del codice secondo due diversi linguaggi di programmazione. Questa domanda porta gli studenti ad esplorare diversi approcci di risoluzione del problema passando da un linguaggio orientato agli oggetti ad uno di natura procedurale.

Elenia Greppi 0512103544
Carmine Cristian Cruoglio 0522501016
Maria Giovanna Albanese 0522501356
Paolo Panico 0522501065