Università degli studi di Torino
Dipartimento di informatica

Progetto di Metodologie didattiche per l'informatica

Attività didattiche inerenti strategie di ricerca

Autore:
Leonardo Magliolo

Anno Accademico 2023/2024

Livello di scuola, classe/i, indirizzo:

L'attività è rivolta principalmente a studenti di scuole secondarie di secondo grado frequentanti il primo biennio dei seguenti indirizzi:

- Licei:
  - Liceo scientifico, articolazione scienze applicate:
    disciplina: Informatica
- Tecnici:
  - Settore economico:
    disciplina: Informatica
  - Settore tecnologico (tutti gli indirizzi):
    disciplina: Tecnologie informatiche

Motivazione e finalità:

L'attività su carta consiste nello svolgimento di un gioco uno contro uno simile a battaglia navale, la cui finalità consiste nell'introdurre agli studenti all'algoritmo di ricerca binaria.

L'attività di programmazione verrà organizzata mediante metodologia PRIMM con linguaggio Python, l'esercitazione guiderà lo studente ad implementare autonomamente l'algoritmo di ricerca binaria su di un array ordinato.

Le ragioni per cui è stato scelto di proporre le seguenti attività sono molteplici:

- Le attività risultano essere adatte al biennio di più indirizzi di scuole secondarie di secondo grado
- L'attività su carta è stata concepita ispirandosi alla teoria sociale del costruttivismo: <<Secondo la teoria del costruttivismo sociale, l'apprendimento avviene principalmente attraverso l'interazione sociale con gli altri, come un insegnante o i compagni>> (citazione: Cambridge Assessment, Apprendimento attivo)
- La durata dell'attività su carta e dell'attività di programmazione è ridotta: l'ipotesi è che l'attività in classe possa essere svolta in circa un'ora e mezza, e che l'attività di programmazione possa esser svolta in meno di un'ora.
- I materiali richiesti per l'attività su carta sono ridotti al minimo indispensabile, è sufficiente un foglio di carta a quadretti e una penna
- Non richiede eccessive conoscenze o abilità pregresse
- Le regole del gioco sono piuttosto simili a quelle del celebre gioco 'battaglia navale', gli studenti non avranno bisogno di eccessive quantità di istruzioni per comprendere il gioco

- Le regole del gioco sono semplici: gli studenti si concentreranno sugli elementi fondamentali caratterizzanti l'esercitazione
- Gli studenti sono tenuti a riflettere sulla strategia che hanno adottato durante lo svolgimento della partita per svolgere l'attività di programmazione: enfasi sulla metacognizione

Prerequisiti:

Attività su carta:
Sarebbe opportuno aver introdotto precedentemente il concetto di ricerca all'interno di una struttura dati

Attività di programmazione in Python:
Dimestichezza con l'utilizzo di:

- Variabili
- Array
- Costrutto if-else
- Iterazioni mediante while
- Funzioni

Contenuti:

Le attività presentate mirano a introdurre i vantaggi relativi la ricerca all'interno di una struttura dati ordinata: per mezzo della strategia "Divide et impera" è possibile ridurre considerevolmente lo spazio di ricerca, minimizzando il numero di operazioni necessarie per effettuare la ricerca.
Il gioco presentato a lezione farà emergere la necessità d'adozione di un algoritmo a ricerca binaria, la cui applicazione è affine in numerosi ambiti dell'informatica.

Traguardi e obiettivi:

Con riferimento ai particolari indirizzi di scuola secondaria superiore elencati in precedenza nella sezione 'Livello di scuola, classe/i indirizzo'

1)Licei:

1.1) Liceo scientifico, articolazione scienze applicate:

Disciplina: Informatica
Area tematica: Algoritmi e linguaggi
Obiettivi preposti (abilità): "Implementazione di un algoritmo in pseudo-codice o in un linguaggio".
Evidenzio inoltre come, secondo la mia opinione, le attività possano esser affini alla piena comprensione di un ulteriore obiettivo preposto riguardo le conoscenze da apprendere in ambito sistemi

operativi: "caratteristiche dei sistemi operativi, definizione di processo e gestione della memoria e file system"

2)Tecnici:

2.1) Settore economico:

Disciplina: Informatica

Conoscenze e abilità: "Organizzazione logica dei dati" (attività unplugged e attività di programmazione), "Fondamenti di programmazione e sviluppo di semplici programmi in un linguaggio a scelta" (attività di programmazione)

2.2) Settore tecnologico (tutti gli indirizzi):

Disciplina: Tecnologie informatiche

Conoscenze: "Fondamenti di programmazione"

Abilità: "Impostare e risolvere problemi utilizzando un linguaggio di programmazione"

(*Tassonomia di bloom)

[Fonti: Indicazioni nazionali MIUR]

Materiali e strumenti:

Attività su carta:

E' sufficiente un foglio di carta, preferibilmente a quadretti.

Attività di programmazione:

E' sufficiente un computer che esegua un semplice editor di testo e un interprete Python.

Linguaggio:

Vengono indicati di seguito alcuni pro e contro dell'utilizzo di Python circa lo svolgimento della specifica attività di programmazione:

Pro:

- La chiarezza della sintassi di Python agevola lo svolgimento del laboratorio, permettendo agli studenti di concentrarsi principalmente sulla logica dell'algoritmo di ricerca binaria.
- Non è richiesta alcuna familiarità pregressa con la programmazione orientata agli oggetti per comprendere appieno i costrutti necessari per completare l'esercitazione, a differenza di quanto avviene con altri linguaggi come Java. Ciò implica un minor carico cognitivo necessario per lo svolgimento della prova da parte dello studente.

Contro:

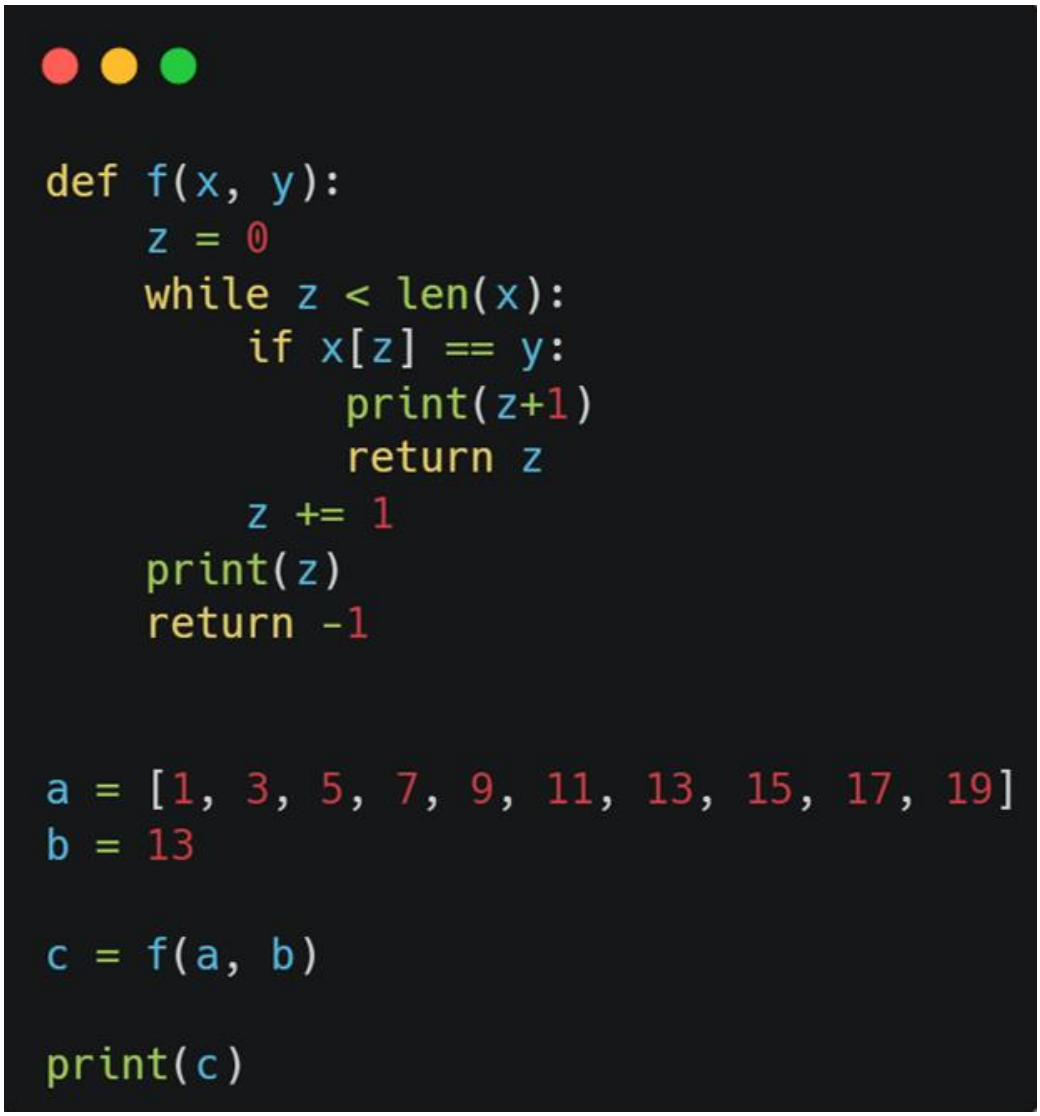
- Gestione dell'indice e dei tipi di dati: Gli studenti potrebbero non ottenere una comprensione approfondita della gestione degli indici e dei tipi di dati utilizzando Python, poiché il linguaggio gestisce molte di queste operazioni in modo poco trasparente rispetto ad altri linguaggi di programmazione.

Sebbene la scelta del linguaggio Python non sia esente da problematiche, si ritiene che per questa specifica esercitazione i suoi contro non inficino particolarmente sulla comprensione dell'argomento.

Sviluppo contenuti:

1 – Materiale didattico per studenti (attività di programmazione):

Fase 1:



```
def f(x, y):
    z = 0
    while z < len(x):
        if x[z] == y:
            print(z+1)
            return z
        z += 1
    print(z)
    return -1


a = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
b = 13

c = f(a, b)

print(c)
```

1) Quali output ti aspetti che il programma stampi? Scrivilo di seguito:

Fase 2:

2) Scarica il codice ed eseguilo. L'output predetto è lo stesso che è stato prodotto?
Modifica il contenuto della variabile b ponendola pari a "19", "1" e "16".
Per ciascuna modifica esegui il codice ed osservane i risultati.

Fase 3:

3)Quando z vale 0 quanto valgono x[z] e y?
Quando z vale 6 quanto valgono x[z] e y?
Scrivilo di seguito:

4)Assegna dei nomi alle variabili in modo tale che abbiano un significato rispetto lo scopo del programma, per ciascuna variabile scrivi di seguito un nome appropriato:

Fase 4:

5) Modifica il codice fornito al punto uno in modo tale che la ricerca del valore all'interno dell'array ordinato parta dall'elemento centrale (nel caso in cui il numero di valori contenuti sia pari è sufficiente approssimare l'indice centrale calcolato per difetto).
Esegui il codice modificato. Il secondo valore stampato è il medesimo? Se sì passa alla richiesta successiva, altrimenti considera delle alternative alle modifiche apportate in maniera tale che l'errore venga risolto.

6) Come mai il primo valore stampato differisce dal codice non modificato?
Riporta la risposta di seguito:

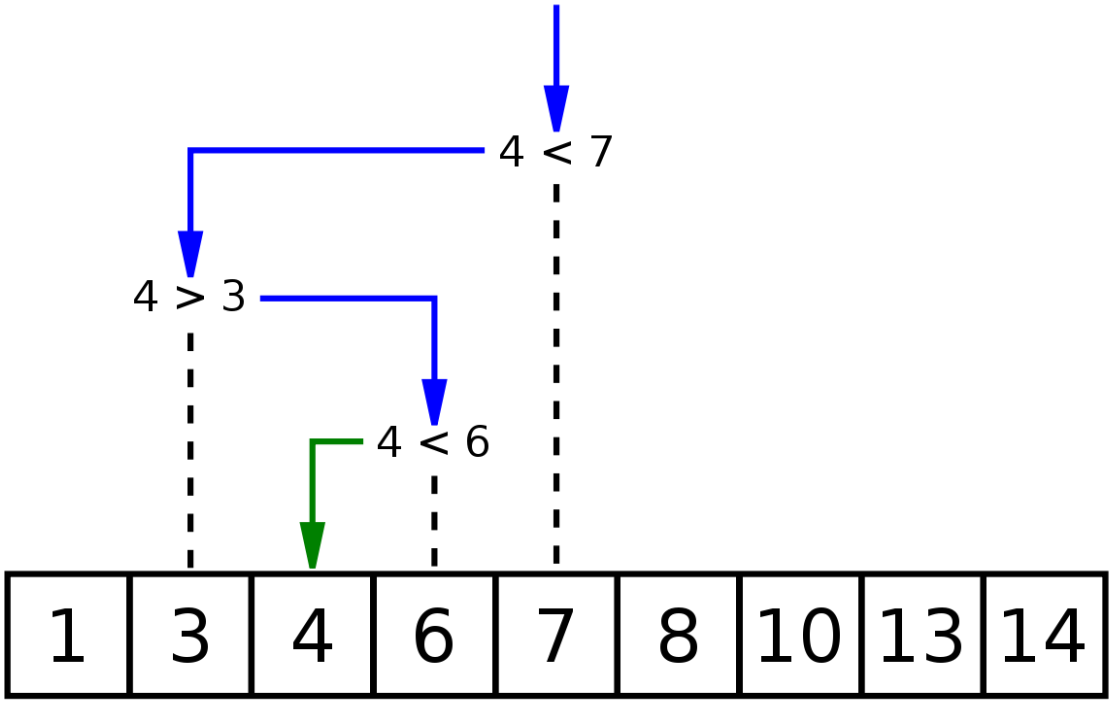
7) Riporta il codice modificato di seguito:

Fase 5:

8) Considera la seguente definizione del metodo di ricerca binaria:
"La ricerca binaria è un metodo per trovare un elemento in un elenco ordinato. Invece di cercare uno per uno dall'inizio alla fine, questo metodo divide

l'elenco a metà ogni volta. Comincia con il punto medio dell'elenco e confronta il valore desiderato con quello nel mezzo. Se sono uguali, hai trovato l'elemento. Se il valore desiderato è più piccolo del valore al centro, guarda solo nella metà inferiore dell'elenco. Se è più grande, guarda solo nella metà superiore. Continua a fare questo taglio a metà fino a quando trovi l'elemento desiderato o fino a quando l'elenco diventa troppo piccolo da esaminare. In questo modo, la ricerca binaria è più efficiente rispetto a una ricerca lineare perché riduce il numero di confronti necessari."

Viene riportata di seguito un'immagine raffigurante un esempio di ricerca binaria in cui l'algoritmo impiega tre confronti per trovare la posizione del valore '4' all'interno di un array ordinato:



Implementa mediante codice il metodo di ricerca binaria illustrato in precedenza in maniera tale che restituisca la posizione di un generico elemento contenuto all'interno di un array contenente numeri interi ordinati (il metodo deve funzionare per qualsiasi dimensione dell'array), nel caso in cui l'elemento richiesto non sia contenuto all'interno dell'array il valore da restituire è '-1'.
Riporta di seguito il codice implementato:

2 – Guida per insegnanti (attività su carta):

Fase 1:
L'insegnante sarà tenuto a spiegare le regole del gioco alla classe in forma orale, successivamente illustrerà alla classe un esempio di partita in chiaro giocata tra due giocatori fittizi. Al termine dell'esempio chiederà agli alunni se sono necessari ulteriori chiarimenti, dopodiché gli studenti verranno suddivisi in gruppi di 4 persone ciascuno (quando possibile).

Consegna:
L'attività su carta consiste nello svolgimento di un gioco uno contro uno simile a battaglia navale.
Ogni giocatore posiziona su di una griglia 20x20 un simbolo che il gioco indica come "il tesoro".
La posizione del tesoro deve esser segreta, ad ogni turno un giocatore cerca di indovinare la posizione del tesoro avversario, qualora il giocatore sbagli ad indovinare l'avversario è costretto a comunicare se:

1)Il tesoro si trova:
- Alla stessa altezza della casella indicata
- Al di sotto della casella indicata
- Al di sopra della casella indicata

2)Il tesoro si trova:
- All'interno della stessa colonna della casella indicata
- A destra della casella indicata
- A sinistra della casella indicata

Vince colui che trova il tesoro nel minor numero di turni: nel caso in cui i due giocatori trovino il tesoro nella stessa quantità di turni l'esito del gioco è pareggio.

Fase 2: Sfida
Coppie di studenti appartenenti allo stesso gruppo giocano simultaneamente, avendo cura di non giocare ogni partita sempre contro lo stesso membro del gruppo.
Durante questa fase l'insegnante assumerà il ruolo di supervisore: è importante che gli studenti siano in grado di sperimentare le possibili strategie senza che gli venga suggerita la strategia di ricerca binaria.

La fase due si conclude dopo circa 20 minuti di gioco, approssimativamente il tempo necessario affinché gli studenti abbiano sperimentato una certa quantità di strategie durante le partite giocate.

Fase 3: Rielaborazione
Ciascun gruppo di studenti è tenuto a riflettere sulle strategie adottate nel corso delle partite giocate.
E' importante che gli studenti descrivano in maniera precisa quali strategie di ricerca secondo la loro esperienza siano state più efficaci e quali proprietà queste strategie avevano in comune.
La fase 3 termina dopo circa 15 minuti.

Fase 4: Discussione
Viene estratto a sorte un membro da ciascun gruppo in modo tale che presenti alla classe i risultati ottenuti in fase di rielaborazione.
Viene concesso un tempo massimo di circa 5 minuti a ciascuno.

Fase 5: Conclusione
Durante la fase di conclusione l'insegnante porrà enfasi sulle proprietà che le varie strategie vincenti posseggono in comune:

- Le strategie vincenti sfruttano l'informazione fornita dall'avversario usando le relazioni d'ordine per ridurre lo spazio di ricerca all'iterazione successiva
- Le strategie vincenti mantengono traccia dei valori scartati in precedenza
- Le strategie vincenti sono più efficienti nello scegliere caselle che potenzialmente sono in grado di scartare un numero maggiore di alternative

Per ciascun punto descritto in precedenza il professore farà degli esempi alla lavagna in relazione al gioco presentato a lezione.
Il professore provvederà successivamente ad introdurre l'algoritmo di ricerca binaria per mezzo di alcuni esempi rappresentati (mediante lavagna) rispetto la sua applicazione su un array ordinato (senza presentare il codice dell'algoritmo nel caso si intenda successivamente svolgere la seconda attività didattica di programmazione).
Il professore concluderà la fase facendo notare come gli stessi punti individuati per le strategie vincenti nel gioco valgano anche nel caso della ricerca ordinata all'interno dell'array.

Snodi e indicatori (inerenti la fase di gioco):

Snodi	Indicatori
Le strategie vincenti mantengono traccia dei valori scartati in precedenza	Gli studenti si annotano (mediante schemi o mediante rappresentazioni numeriche) lo spazio di ricerca scartato a partire dai turni precedenti
Le strategie vincenti sfruttano l'informazione fornita dall'avversario usando le relazioni d'ordine per ridurre lo spazio di ricerca all'iterazione successiva	Gli studenti scartano regioni contigue all'interno dello spazio di ricerca (mediante rappresentazione schematica o numerica)
Le strategie vincenti sono più efficienti nello scegliere caselle che potenzialmente sono in grado di scartare un numero maggiore di alternative	Gli studenti scelgono caselle del campo avversario che sono posizionate al centro rispetto lo spazio di ricerca ammissibile

2- Attività di programmazione:

È stata scelta la metodologia PRIMM per progettare l'attività di programmazione. Quest'ultima è individuale e può essere completamente svolta al computer.
Durante l'esercitazione il ruolo dell'insegnate è quello del facilitatore, si rende inoltre disponibile per ulteriori chiarimenti ed osserva l'operato degli alunni durante lo svolgimento della prova.

Fase 1 (Prediction):
Durante la prima fase, si presenta un programma (i cui nomi delle variabili devono essere non rappresentativi) che effettui la ricerca di una chiave utilizzando un approccio lineare su di un array ordinato di valori interi. L'esecuzione del codice si conclude con la stampa dell'indice relativo al valore cercato all'interno dell'array. Si chiede agli studenti di predire l'output del codice e dopo quante iterazioni l'algoritmo termina.

Fase 2 (Run):
Viene chiesto allo studente di eseguire il codice e di osservare se l'output del programma è come predetto al punto precedente.

Viene chiesto allo studente di variare il contenuto della variabile che identifica la chiave da cercare dentro l'array con determinati valori contenuti dentro l'array ordinato,
e dopo ogni modifica viene chiesto allo studente di eseguire il programma ed osservarne l'output.

Fase 3 (Investigate):
Viene chiesto allo studente di effettuare il debug del codice presentato in fase di prediction in modo tale che possa rispondere ad alcune domande circa lo stato di determinate variabili all'i-esima iterazione dell'algoritmo.
Viene inoltre richiesto allo studente di assegnare un nome a ciascuna variabile del frammento di codice proposto.

Fase 4 (Modify):
Viene richiesto allo studente di modificare il codice presentato durante la prima fase in maniera tale che la ricerca dell'array avvenga a partire dal valore centrale dell'array (richiedendo che la modifica debba esser attuabile anche qualora l'array contenga un numero pari di elementi).
Viene chiesto allo studente di verificare che l'output ottenuto sia lo stesso del codice precedente, di verificare che il numero di iterazioni rispetto al caso precedente sia ridotto (richiedendo anche motivazioni a riguardo le cause che hanno ridotto i tempi d'esecuzione).

Fase 5 (Make):
Vengono mostrati agli studenti degli esempi di esecuzione (mediante approccio grafico) dell'algoritmo di ricerca binaria, accompagnati da una breve descrizione testuale dello stesso e viene richiesto loro di scrivere un programma che sia in grado di effettuare tale ricerca per ogni array ordinato contenente interi.

Snodi	Indicatori	Fase
Lo studente comprende il flusso d'esecuzione del programma	Lo studente indica come risposta: "7 6" all'interno del quesito 1)	Prediction
Lo studente comprende quale funzione assolve il programma	Lo studente associa i seguenti nomi (o sinonimi) alle variabili: • (f, ricerca) • (x, array) • (a, array)	Investigate

	• (b, chiave) • (y, chiave) • (z, indice) • (c, indice_chiave)	
Lo studente comprende che prima della ricerca deve essere effettuata una suddivisione tra intervalli	Lo studente inserisce all'interno della funzione di ricerca un costrutto if-else che esegua un controllo sul valore di un operatore di ordinamento (Es: if array[len(array)/2] > chiave)	Modify
Lo studente riconosce le ragioni che permettono alla versione modificata d'esser più efficiente	Lo menzionano all'interno della spiegazione fornita al punto 6)	Modify
Lo studente comprende la logica di suddivisione a sottointervalli adottata nell'algoritmo di ricerca binaria	Lo studente adopera la strategia suggerita in fase Modify ad ogni iterazione dell'algoritmo	Make

3 - Indicazioni per la valutazione dell'apprendimento:

Durante lo svolgimento delle attività unplugged e di programmazione è sufficiente che l'insegnante si assicuri di tenere a mente snodi e indicatori indicati all'interno della relazione in maniera tale che possa adattare la sua spiegazione in relazione alle esigenze della classe: svolgere le attività senza assicurarsi che gli studenti abbiano superato gli snodi necessari le rende poco efficaci.

- Esercizi di programmazione:
  Tipi di scopo esercizi mediante block model:

Numero esercizio	x-axis	y-axis
1	Program Execution	Macrostructure
2	Program Execution	Macrostructure
3	Program Execution	Blocks
4	Function	Macrostructure
5	Function	Blocks

6	Function	Atoms
7	Function	Blocks
8	Function	Macrostructure

- Misconceptions associate allo svolgimento dell'attività di programmazione *****:

No. Misconception	Argomento	Descrizione testuale	Riferimento a caso specifico
1	Generale	Il computer conosce l'intenzione del programma o di un pezzo di codice e agisce di conseguenza.	Esercizio 5): Lo studente non considera che è necessario effettuare un controllo circa l'intervallo in cui cercare il valore chiave in quanto crede implicitamente che l'algoritmo se ne occupi autonomamente
3	Generale	I valori vengono aggiornati automaticamente in base a un contesto logico.	Esercizio 8): Durante la ricerca lo studente non aggiorna il valore dell'indice relativo la prossima ricerca all'interno dell'array, effettuando sempre lo stesso controllo per ogni iterazione del while: ciò renderà impossibile la terminazione del programma eseguito
/	/	Lo studente non tiene traccia di valori necessari al compimento della funzione relativa l'algoritmo	Esercizio 8): Lo studente non tiene traccia degli indici relativi l'inizio della sotto-lista e la fine della stessa. Per tanto sbaglia a calcolare l'elemento centrale della sotto-lista corretta. Qualora calcoli sempre la stessa divisione sarebbe impossibile giungere alla terminazione del programma
/	/	Lo studente non riesce a determinare le	Esercizio 8): Lo studente non comprende che se la chiave è maggiore dell'estremo

		condizioni di terminazione con fallimento	superiore dell'intervallo selezionato o minore dell'estremo inferiore dell'intervallo è necessario che la ricerca concluda con fallimento
43	Chiamate a funzione	Confusione sulla posizione dei valori di ritorno	Esercizi 1), 3), 5) e 8): L'allievo non comprende che quando viene ritornato un valore termina l'esecuzione della porzione di codice relativa la funzione chiamata, pertanto commette errori (che fatica a correggere) causati dalla posizione dei return nel codice. Alternativamente nell'esercizio 1): L'allievo non comprende che è possibile usare più di un return all'interno della stessa funzione
151	Misc	Confusione tra un array e la sua cella	Esercizi 1), 3), 5), 8): Lo studente non comprende che il risultato dell'istruzione x[z] è un tipo di dato intero e non si tratta di un array
/	/	Confusione tra operatori di divisione intera e con virgola	Esercizi 5), 8): Lo studente scambia l'operatore di divisione intera (//) con l'operatore di divisione con virgola (/). Il risultato della divisione sarà un numero non intero e durante l'accesso ai valori di z l'interprete Python restituirà il seguente errore: TypeError: list indices must

			be integers or slices, not float

(*Tratte da "Visual Programming Simulation in Introductory Programming Education" Autore: Juha Sorva, pagina 358)

Non sono state inserite tutte le misconceptions relative argomenti che si presumono esser già consolidati. Qualora si verifichino è possibile consultare il testo citato per ulteriori approfondimenti.

Rubrica valutativa per l'attività su carta:

	Assente	**Parziale**	**Adeguata**
Individuazione di una strategia efficiente per la ricerca	Lo studente adopera strategie casuali che non sfruttano le informazioni fornite dall'avversario	Lo studente adopera strategie che sfruttano le informazioni fornite dall'avversario senza che lo spazio delle ipotesi ammissibili venga partizionato in maniera ottimale	Lo studente adopera strategie che sfruttano le informazioni fornite dell'avversario in maniera tale che massimizzino la probabilità di ridurre il numero di turni prima di trovare il tesoro
Collaborazione con il gruppo in fase di rielaborazione	Lo studente non si confronta con il gruppo in fase di rielaborazione	Lo studente si confronta passivamente con il gruppo, la sua esperienza non è parte attiva del confronto	Lo studente si confronta attivamente con il gruppo, cerca di esporre attivamente le ragioni che giustificano le sue strategie prendendo in considerazione anche le strategie alternative esposte dagli altri membri del gruppo

Chiarezza espositiva in fase di discussione	Lo studente non è in grado di descrivere alla classe la sua strategia, né tantomeno è in grado di giustificarne l'efficacia	Lo studente è in grado di descrivere alla classe la sua strategia, ma non è in grado di evidenziare le cause che la rendono efficace	Lo studente è in grado di descrivere la sua strategia ed è in grado di evidenziare le cause che la rendono efficace. Lo studente è in grado di comparare le sue strategie con altre strategie

Rubrica valutativa per l'attività di programmazione:

	Assente	**Parziale**	**Adeguata**
Correttezza del codice prodotto	Il codice prodotto dallo studente non funziona per la maggioranza degli input	Il codice prodotto dallo studente funziona per la maggioranza degli input, ma non funziona per alcuni casi limite	Il codice prodotto dallo studente funziona per qualsiasi input
Leggibilità del codice	Il codice prodotto dallo studente risulta scarsamente comprensibile (es: nomi delle variabili non rappresentativi, indentazione inadeguata, eccetera)	Il codice prodotto dallo studente risulta comprensibile nella sua globalità sebbene ne esistano alcune porzioni scarsamente comprensibili	Il codice prodotto dallo studente risulta comprensibile nella sua globalità
Efficienza del codice prodotto	Il codice prodotto dallo studente esegue più iterazioni di quante necessarie allo svolgimento del	Il codice prodotto dallo studente esegue il numero minimo di iterazioni necessarie a svolgere il compito	Il codice prodotto dallo studente esegue il numero minimo di iterazioni necessarie a svolgere il compito

	compito. Il codice dello studente contiene inoltre istruzioni inutili al fine di svolgere il compito	sebbene contenga alcune istruzioni inutili	e non contiene istruzioni inutili
Comprensione del problema	Nelle domande a risposta aperta lo studente non riesce a giustificare i risultati prodotti dall'algoritmo	Nelle domande a risposta aperta lo studente giustifica parzialmente i risultati prodotti dall'algoritmo	Nelle domande a risposta aperta lo studente giustifica con precisione i risultati prodotti dall'algoritmo