Algoritmo: Strategia Greedy per la Selezione degli Iscritti

Introduzione:
La strategia greedy è un paradigma di risoluzione di problemi che fa scelte localmente ottimali in ogni passo con l'obiettivo di ottenere una soluzione globalmente ottimale. Nel contesto della selezione degli iscritti a un evento, la strategia greedy può essere applicata per massimizzare l'efficienza e soddisfare requisiti specifici come il numero massimo di partecipanti e la considerazione del livello di esperienza degli utenti.

Pseudocodice:
L'algoritmo `selezionaIscritti` qui di seguito implementa una strategia greedy per la selezione degli iscritti a un evento, tenendo conto del numero massimo di partecipanti e del livello degli utenti. L'approccio si basa sull'idea di selezionare in modo iterativo gli utenti con il livello più adatto, iniziando dal livello dell'evento e ampliando la ricerca ai livelli adiacenti solo se necessario.

pseudocodice.txt

```
@metodo di EventManager

algoritmo selezionaIscritti(Lista listaIscritti)
    S <- {}
    limiteMax = Event.maxPeople

    if (listaIscritti.dim <= limiteMax) then S <- all(listaIscritti)
    else
        livello = Event.level
        int i = 0  //posti occupati
        int j = 0  //distanza dal livello evento

        while((i != limiteMax) and (listaIscritti != {})) do
            utenteConfermato = seleziona(listaIscritti, livello, j)
            if(not listaIscritti.contains(anotherUser con userLevel = livello +- j)) then j+1
            listaIscritti <- listaIscritti / {utenteConfermato}
            S <- S U {utenteConfermato}
            i + 1

    return S

//prima vengono considerati gli utenti con lo stesso livello dell'evento
//poi quelli ai livelli +-1, +-2 fino a esaurimento posti

funzione seleziona(listaIscritti, livello, distanza)
    return (User con userLevel = livello + distanza) or (User con userLevel - distanza)
```

Questo algoritmo mira a massimizzare la partecipazione all'evento in modo efficiente, seguendo una logica greedy nell'assegnazione dei posti agli iscritti. La selezione degli utenti avviene in modo iterativo, considerando prima gli utenti con il livello esatto dell'evento e successivamente estendendo la ricerca ai livelli adiacenti solo se necessario.

Implementazione in Java:

Ecco un'implementazione in Java dell'algoritmo descritto nello pseudocodice. Questa implementazione utilizza una classe `User` e una classe `Event` per rappresentare gli utenti

e gli eventi, rispettivamente. La funzione `selezionaIscritti` restituisce l'insieme di utenti selezionati per partecipare all'evento.

```
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class EventSelection {

    public Set<User> selezionaIscritti(List<User> listaIscritti, Event event) {
        Set<User> S = new HashSet<>();
        int limiteMax = event.getMaxPeople();

        if (listaIscritti.size() <= limiteMax) {
            S.addAll(listaIscritti);
        } else {
            int livello = event.getLevel();
            int i = 0;  // Numero di posti occupati
            int j = 0;  // Distanza dal livello dell'evento

            while ((i != limiteMax) && (!listaIscritti.isEmpty())) {
                User utenteConfermato = seleziona(listaIscritti, livello, j);

                if (!listaIscritti.contains(new User(livello + j)) &&
                    !listaIscritti.contains(new User(livello - j))) {
                    j++;
                }

                listaIscritti.remove(utenteConfermato);
                S.add(utenteConfermato);
                i++;
            }
        }

        return S;
    }

    private User seleziona(List<User> listaIscritti, int livello, int distanza) {
        // Restituisci un utente con un livello pari a livello + distanza o livello - distanza
        return new User(livello + distanza);
    }
}
```

Analisi di Complessità e Costo Computazionale:

Complessità Temporale:
L'implementazione proposta ha una complessità temporale che dipende dal numero massimo di partecipanti all'evento (`limiteMax`) e dalla dimensione della lista degli iscritti (`listaIscritti`). Nel caso peggiore, dove è necessario iterare su tutti gli iscritti, la complessità temporale è O(limiteMax * n), dove n è la dimensione della lista degli iscritti.

Complessità Spaziale:
La complessità spaziale è dominata dalla creazione dell'insieme `S`. Nel caso peggiore, dove tutti gli iscritti partecipano all'evento, la complessità spaziale è O(n), dove n è la dimensione della lista degli iscritti. L'uso di un HashSet aiuta a gestire l'insieme senza duplicati.

Questa analisi fornisce una panoramica delle prestazioni dell'algoritmo, ma è importante considerare le specifiche del caso d'uso per un'analisi più dettagliata.