Numbering (`numbering`)

Data una foresta di N nodi, una *numerazione* di essa è un assegnamento di interi positivi, eventualmente ripetuti, ad ogni arco della foresta. Una *numerazione* si dice *bella* se, per ogni nodo, i suoi archi hanno i numeri $1, 2, \ldots, d$ in qualche ordine (dove d è il grado del nodo).

Vengono dati N interi positivi $A_0, \ldots, A_{N-1}$. Determinare se esiste una foresta di N nodi tale che:

- per ogni $0 \leq i \leq N-1$, il grado del nodo i è A_i;
- ammette almeno una numerazione *bella*.

Inoltre, se esiste una tale foresta, costruire un esempio.

Implementazione

Dovrai sottoporre un unico file, con estensione `.cpp`.

> ☞ Tra gli allegati a questo task troverai un template `numbering.cpp` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

C++
```
variant<bool, vector<pair<int, int>>> find_numbering(int N, vector<int> A);
```

- L'intero N rappresenta il numero di nodi.
- L'array A, indicizzato da 0 a $N-1$, contiene i valori A_0, A_1, …, A_{N-1}, dove A_i è il grado del nodo i-esimo.
- La funzione deve restituire un booleano o un array di coppie di interi.
  - Se non esiste alcuna foresta valida (soddisfacente le condizioni del testo), la funzione deve restituire `false`.
  - Se esiste una foresta valida, hai due opzioni:
    - Per ottenere il punteggio pieno, la funzione deve restituire un array di coppie di interi, rappresentante gli archi di una foresta valida.
    - Per ottenere un punteggio parziale, la funzione deve restituire `true` o un array di interi che non descrivono una foresta valida.

Il grader chiamerà la funzione `find_numbering` e stamperà sul file di output:

- Se la funzione restituisce `false`, stamperà una riga contenente la stringa `NO`.
- Se la funzione restituisce `true`, stamperà una riga contenente la stringa `YES`.
- Se la funzione restituisce un array di coppie di interi di lunghezza M, stamperà una riga contenente la stringa `YES`, seguita da una riga contenente M, seguita da M righe contenenti le coppie di interi dell'array.

Grader di prova

La directory relativa a questo problema contiene una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati di input dal file `stdin`, chiama la funzione che dovete implementare e scrive il risultato sul file `stdout`.

Il file di input è composto da 2 righe, contenenti:

- Riga 1: l'intero N.
- Riga 2: gli N interi $A_0, A_1, \ldots, A_{N-1}$.

Il file di output è composto da più righe, contenenti i valori restituiti dalla funzione `find_numbering`.

Assunzioni

- $2 \leq N \leq 10^5$.
- $0 \leq A_i \leq N - 1$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Il punteggio associato ad un subtask è il minimo dei punteggi ottenuti nei singoli test case.

- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [16 punti]**: $A_i \leq 2$.
- **Subtask 3 [12 punti]**: $A_i \leq 3$.
- **Subtask 4 [16 punti]**: Poniamo $\texttt{count}(i)$ pari al numero di occorrenze di i in A. È garantito che $\texttt{count}(i) \geq \texttt{count}(i+1) + \texttt{count}(i+2) + \ldots$ per ogni $1 \leq i \leq N - 1$.
- **Subtask 5 [10 punti]**: $N \leq 12$.
- **Subtask 6 [24 punti]**: $N \leq 500$.
- **Subtask 7 [22 punti]**: Nessuna limitazione specifica.

Per ogni test case in cui esiste una foresta valida, la tua soluzione:

- ottiene il punteggio pieno se restituisce una foresta valida.
- ottiene il 50% dei punti se restituisce `true` o un array che non descrive una foresta valida.
- ottiene 0 punti altrimenti.

Per ogni test case in cui non esiste una foresta valida, la tua soluzione:

- ottiene il punteggio pieno se restituisce `false`.
- ottiene 0 punti altrimenti.

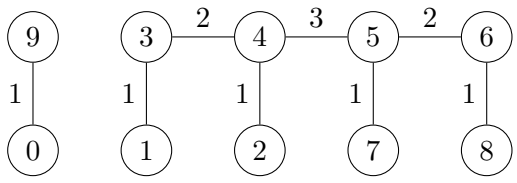
Esempi di input/output

stdin	stdout
4 1 1 2 1	NO
10 1 1 1 2 3 3 2 1 1 1	YES 8 0 9 1 3 2 4 3 4 4 5 5 6 5 7 6 8

Spiegazione

Nel **primo caso d'esempio**, vogliamo una foresta valida con 4 nodi: 3 con grado 1 e 1 con grado 2. Possiamo mostrare che questo non è possibile. Supponiamo che una tale foresta esista, allora dovrebbe esistere un arco con numero 2 uscente dal nodo con grado 2. Questo arco si connette ad un altro nodo che dovrebbe avere grado almeno 2. Tuttavia, tale nodo non esiste, poiché tutti gli altri nodi hanno grado 1.

Nel **secondo caso d'esempio**, vogliamo una foresta valida con 10 nodi: 6 con grado 1, 2 con grado 2 e 2 con grado 3. Tale foresta esiste e l'output è rappresentato di seguito:



Nota che i nodi 4 e 5 hanno tre archi etichettati $1, 2$ e 3.
Inoltre i nodi 3 e 6 hanno due archi etichettati 1 e 2.
Infine i nodi $0, 1, 2, 7, 8$ e 9 hanno un arco etichettato 1.