#include <assert.h>

#include <ctype.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\* ltrim(char\*);

char\* rtrim(char\*);

char\*\* split\_string(char\*);

// Complete the balancedSums function below.

/\*

\* To return the string from the function, you should either do static allocation or dynamic allocation

\*

\* For example,

\* char\* return\_string\_using\_static\_allocation() {

\* static char s[] = "static allocation of string";

\*

\* return s;

\* }

\*

\* char\* return\_string\_using\_dynamic\_allocation() {

\* char\* s = malloc(100 \* sizeof(char));

\*

\* s = "dynamic allocation of string";

\*

\* return s;

\* }

\*

\*/

static char str1[] = "YES";

static char str2[] = "NO";

char\* balancedSums(int arr\_count, int\* arr) {

if(arr\_count==1)

{

return str1;

}

//////

else if(arr\_count ==2)

{

return str2;

}

/////

///

else {

int i,j;

int sum1,sum2;

int sum;

sum = 0;

for(i=0;i<arr\_count;i++)

{

sum = sum + arr[i];

}

sum1 = 0;

for(i=0;i<arr\_count;i++)

{

//sum2=0;

if(i==0)

{

sum1=0;

}else {

sum1 = sum1 + arr[i-1];

}

/\* for(j=0;j<i;j++)

{

sum1 = sum1 + arr[j];

}

for(j=i+1;j<arr\_count;j++)

{

sum2 = sum2 + arr[j];

}

\*/

if(sum1 == sum-sum1-arr[i])

{

return str1;

}

}

return str2;

}

}

int main()

{

FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

char\* T\_endptr;

char\* T\_str = ltrim(rtrim(readline()));

int T = strtol(T\_str, &T\_endptr, 10);

if (T\_endptr == T\_str || \*T\_endptr != '\0') { exit(EXIT\_FAILURE); }

for (int T\_itr = 0; T\_itr < T; T\_itr++) {

char\* n\_endptr;

char\* n\_str = ltrim(rtrim(readline()));

int n = strtol(n\_str, &n\_endptr, 10);

if (n\_endptr == n\_str || \*n\_endptr != '\0') { exit(EXIT\_FAILURE); }

char\*\* arr\_temp = split\_string(rtrim(readline()));

int\* arr = malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

char\* arr\_item\_endptr;

char\* arr\_item\_str = \*(arr\_temp + i);

int arr\_item = strtol(arr\_item\_str, &arr\_item\_endptr, 10);

if (arr\_item\_endptr == arr\_item\_str || \*arr\_item\_endptr != '\0') { exit(EXIT\_FAILURE); }

\*(arr + i) = arr\_item;

}

int arr\_count = n;

char\* result = balancedSums(arr\_count, arr);

fprintf(fptr, "%s\n", result);

}

fclose(fptr);

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) {

break;

}

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') {

break;

}

alloc\_length <<= 1;

data = realloc(data, alloc\_length);

if (!data) {

data = '\0';

break;

}

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

data = realloc(data, data\_length);

if (!data) {

data = '\0';

}

} else {

data = realloc(data, data\_length + 1);

if (!data) {

data = '\0';

} else {

data[data\_length] = '\0';

}

}

return data;

}

char\* ltrim(char\* str) {

if (!str) {

return '\0';

}

if (!\*str) {

return str;

}

while (\*str != '\0' && isspace(\*str)) {

str++;

}

return str;

}

char\* rtrim(char\* str) {

if (!str) {

return '\0';

}

if (!\*str) {

return str;

}

char\* end = str + strlen(str) - 1;

while (end >= str && isspace(\*end)) {

end--;

}

\*(end + 1) = '\0';

return str;

}

char\*\* split\_string(char\* str) {

char\*\* splits = NULL;

char\* token = strtok(str, " ");

int spaces = 0;

while (token) {

splits = realloc(splits, sizeof(char\*) \* ++spaces);

if (!splits) {

return splits;

}

splits[spaces - 1] = token;

token = strtok(NULL, " ");

}

return splits;

}