//Trie

//Complexity : making a trie : O(S), searching : O(S)

struct node {

bool isEnd;

node \*next[11];

node() {

isEnd = false;

for(int i = 0; i < 10; i++)

next[i] = NULL;

}

};

bool found;

//trie of a string abc, ax

// [start] --> [a] --> [b] --> [c] --> endMark

// |

// [x] --> endMark

//creates trie, returns true if the trie we are creating is a segment of a string

//to only create a trie remove lines which are comment marked

bool create(char str[], int len, node \*current)

{

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

int pos = str[i] - '0';

if(current->next[pos] == NULL)

current->next[pos] = new node();

current = current->next[pos];

if(current->isEnd)

return true;

}

current->isEnd = true;

return false;

}

void del(node \*current)

{

for(int i = 0; i < 10; i++)

if(current->next[i] != NULL)

del(current->next[i]);

delete current;

}

void check(node \*current)

{

for(int i = 0; i < 10; i++) {

if(current->next[i] != NULL)

check(current->next[i]);

}

if(found)

return;

if(current->isEnd && !found) {

for(int i = 0; i < 10 && !found; i++)

if(current->next[i] != NULL) {

found = 1;

}

}

}

int main()

{

//freopen("in", "r", stdin);

//freopen("out", "w", stdout);

int t, n;

char S[15];

scanf("%d", &t);

while(t--) {

found = 0;

node\* root = new node(); //important part of the code

scanf("%d", &n);

while(n--) {

scanf(" %s", S);

if(!found)

if(create(S, strlen(S), root))

found = 1;

}

if(!found)

check(root);

if(found)

printf("NO\n");

else

printf("YES\n");

del(root);

}

return 0;

}