COP 3502

8/25/2023

- ✓ (0) ULAs - Update
- ✓ (1) SLM completion
- ✓ (2) malloc, calloc, realloc
- (3) array of strings
- (4) other examples

Binary Search Review

If an array sorted and we're searching for a value in the array, we can ~~compile~~ compare halfway btw ~~0 &~~ our search endpts + then cut our search space in half

[2, 6, 15, 20, 25, 60, 82] search 60

comp1: 60 > 20

[25, 60, 82]

comp2: found

[2, 6, 15, 20, 25, 60, 82] search 12

comp1: 12 < 20

[2, 6, 15]

comp2: 12 > 6

[15]

12 < 15

not found

Let k = # comp

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$k = \log_2 n$$

SLM edit

① let's say I dynamically allocate an array to be really big but it ends small so I want to free some room...

② my array is full but I want to extend it!

```
arr = realloc(arr, 2*n*sizeof(int))
```

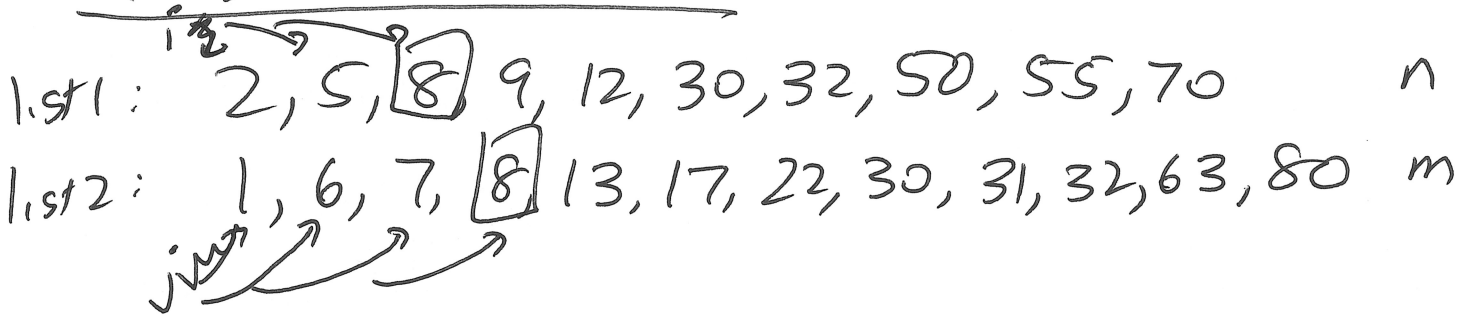
arr: cur ptr; 2*n: new size

realloc(arr, 0) is same as free(arr

Alg 2 for SLM

for each x in list1: n m
run binarysearch(list2, x) lgm or lgn
Run time O(nlgm) or O(mlgn)

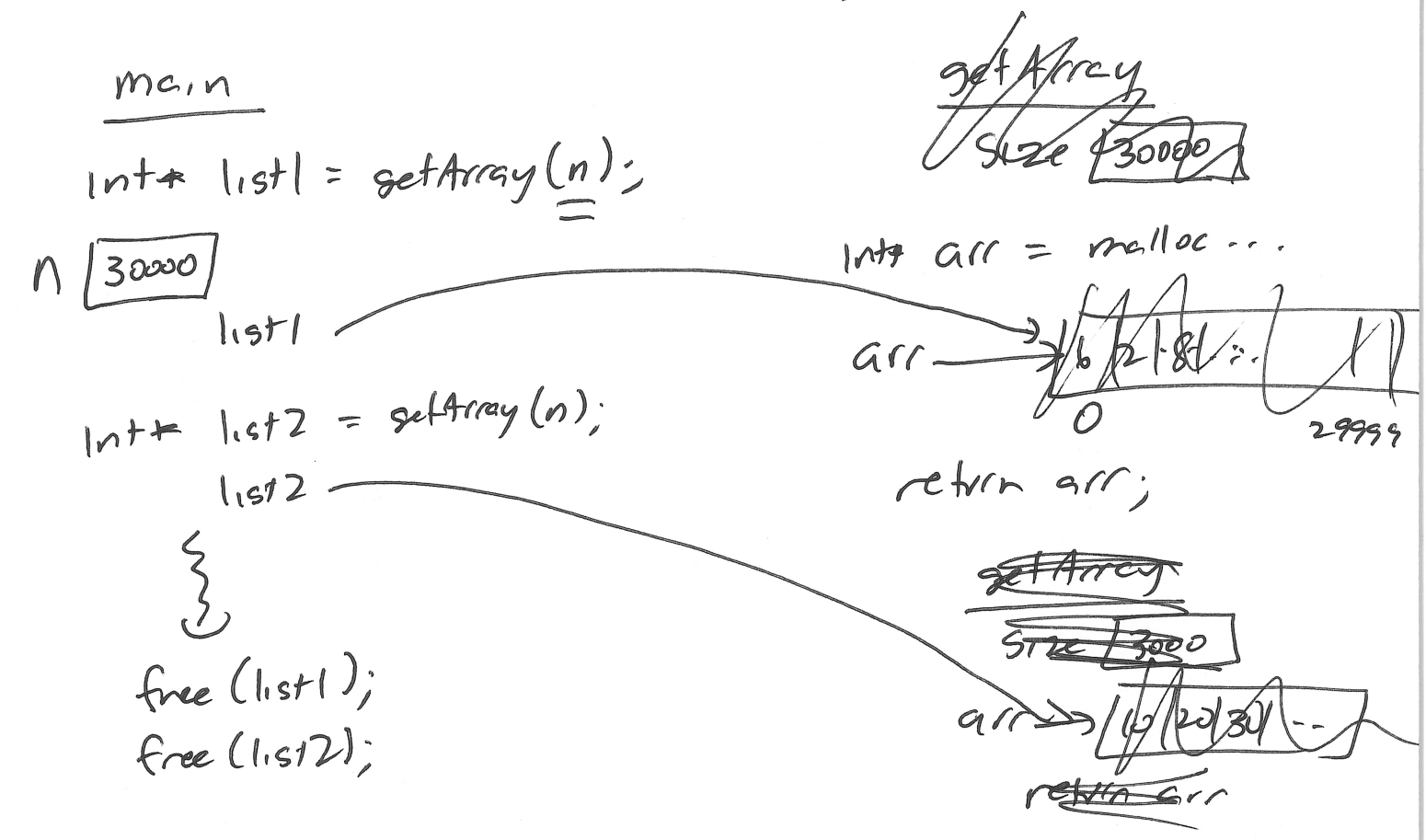
Alg 3 for SLM




```
i=0;
j=0;
while (i<n && j<m) {
    if ( list1[i] < list2[j] ) i++;
    else if (list2[j] < list1[i] j++;
    else {
        printf("%d ", list1[i]);
        i++;
        j++;
    }
}
```

O(n+m) runtime

Picture of SLM code



Calloc

TYPE* array = calloc(num_elems, size of an element)

anytype

```
int* freq = calloc (101, sizeof(int));
```

(1) 2 params (num elems, size of each)

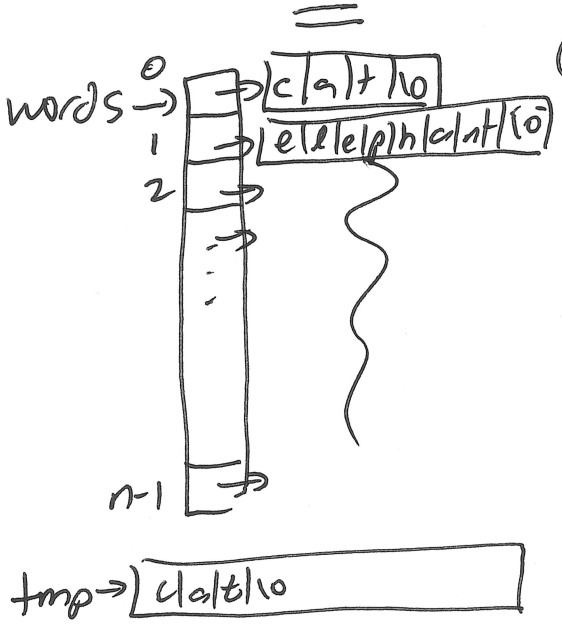
(2) 0's out all memory.

{ Same as a reg array.

Array of Strings

① efficient mem use

② char** words = calloc(n, sizeof(char*));



③ read a word into a tmp string that's big enough.

④ Once we know how much room we need, allocate it for that string

⑤ copy from tmp to words

```
for (int i=0; i<n; i++) {
    scanf("%s", tmp);
    words[i] = calloc(strlen(tmp)+1, sizeof(char));
    strcpy(words[i], tmp);
}
```

Free

```
for(int i=0; i<n; i++)
    free(words[i]);
free(words)
```