AlphaX Design Comments – Photo Search Engine
20 March 2021

Functional Design (Requirements, Use Cases) – 2/2 points

- Good problem summary! Except you talk about selecting a “page” from the list – I think you mean a photo.
- If you want, you can limit the number of tags (to something reasonable like 20)
- It would be nice to have a use case to ADD a new photo plus tags.
- Use case names should be short: Add tags, Delete tags, Search by tags, Display photo, etc.
- Add or delete tag use case: it is not realistic to list all the photos and ask the user to choose (because there will be too many). You could ask the user to enter the name of the image file (or the image title) and do a partial match based on the string. OR you could have the user search by tags first, then select one of the matching items to do add/delete tag.
- Search for photos use case – there’s also an alternative if the user just enters <RETURN> with no tags.
- Display on browser use case – I think you should remove steps 1 and 2. Step 3 should be to SELECT one of the photos from the displayed list. An alternative is that the user does not select any photo, in which case you would go back to the menu.

Overall EXCELLENT use cases!

Architectural Design (Module diagram and table) – 2/2 points

- The Index Builder must update BOTH the file and the hash table when the user adds or removes tags.
- User interface will probably not be graphical... unless you want to use a Java-based front end (which you can do). One advantage would be that this would remove the need to use a browser for image display.

Data Structure/File Design – 3/3 points

- Good level of detail. However, I do not think you need the alphabetic hash table of photos. If you want, you can have a separate hash table keyed by photo name, but it should not use the dictionary structure because the “first letter” hash function is not a good hash function. It will not distribute data items evenly because there are many more words that start with some letters than with others.

I also think you should create a temporary data structure to hold the results of a search or a similarity calculation, rather than having a true/false flag on each photo. Otherwise, you’ll have to look at ALL photos again, after the search. Instead, use a temporary linked list or dynamically allocated array to hold pointers to matched photos that you find during the search.

I am not sure whether you plan to have a “master list” of all photos. I think this will be useful. The hash table items will hold references to items in this master list. Having the master list will simplify writing the output file.

Your file structure seems to assume that all photo names and tags will be single words. This seems a bit restrictive. If you use a delimiter character other than space, you can remove this restriction, e.g.

```
Tabby Cat:3:home/cat.jpg
brown:striped fur:cute tiger
```

Algorithm Design - 2/2 points

- Your pseudocode is a bit too much like code. It assumes the reader has your structs as a reference

- findphoto algorithm
    - o This allows only one photo to be returned. This is okay, as long as you are sure there will be no duplicate names
    - o As noted above, I don't think you should use an initial-char-based hash function, but a more general hash using the title of the photo as the key.
- add tag algorithm
    - o You also need to be sure this gets added to the file. Or else you need to have a function to write all the data before the user exits (which will be easier).
- check tag and search algorithms
    - o This is a case where you can use the count. You will access the hash table once for each tag. For each photo that matches the tag, add 1 to count. As soon as 'count' matches the number of tags in the tag list, you can copy a pointer to the photo to a results list.
- include/exclude tag search
    - o I think you should first do the "included" tag search and get the results list. Then you just go through the results list and remove any photos that also have any "excluded" tags
- similar photos
    - o Excellent idea to use a priority queue! However, you need to increment a counter for each match, because the "priority" will be based on the count. (Max possible count is the total number of tags on the original image)

- Overview – I think you should allow finding similar photos without the user having to display a photo first. I would suggest organizing your system as follows:

-- find photos (by tags – included, or included excluded) – System shows list of results
 --- Ask user to select photo
 --- Ask user what to do with the photo
  ---- display
  ---- find similar
  ---- add/remove tags

After find similar, the user would have the same options with the new list.

(If you don't like my suggestion, you can keep your current organization, but I think my flow would result in less duplicated code as well as being more user friendly.)

Effort, following instructions, etc. - 1/1 point

Great job!

Total grade -- 10/ 10