University of Nevada, Las Vegas Computer Science 477/677 Fall 2022

Assignment 6: Due Saturday November 12, 2022, Midnight

This is the final version.

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. Please follow Mr. Wang's instructions on how to submit your completed assignment.

1. True or False.

   (a) **F** Computers are so fast nowadays that there is no longer any need to be concerned about the time complexity of a progralm.

   (b) **F** Computers have so much memory nowadays that there is no longer any need to be concerned about the space complexity of a program.

2. Your company sells boating accessories. You need to choose a hash function for your set of customer files. Which of the following would be a better hash function?

   (a) The last four digits of the customer's social security number.

   (b) the last four digits of the customer's zip code.

   Why did you make that choice?

   Last 4 digits of the SSN. Zip code is a very bad choice, since customers are concentrated in areas near water.

3. If separate chaining is used to resolve collisions in a hash table of size m which stores n items, the probability that a given place has exactly k items is approximately

$$\frac{(n/m)^k}{k!\, e^{n/m}}$$

   as given by the Poisson distribution. If $m = 100$ and $n = 200$, the average number of items in a place is 2. Approximately how many places will have exactly 2 items? Choose from the following list.
   100
   55
   45
   27
   21
   13

   $\frac{n}{m} = 2$, and $k = 2$. So the number of places that have exactly 2 items is expected to be approximately

$$100 \cdot \frac{2^2}{2!e^2} = \frac{200}{e^2} \approx 27$$

4. You are trying to construct a cuckoo hash table of size 10, where each of the 9 names listed below has the two possible hash values, indicated in the array. Can you construct that table? Construct the table, or show that it can't be done by using Hall's marriage theorem.

	h1	h2
Ann	0	3
Bob	1	3
Ted	6	8
Sue	3	6
Gus	2	7
Cal	4	7
Dan	1	9
Sal	6	9
Eve	5	8

0	Ann
1	~~Bob~~ ~~Dan~~ Bob
2	Gus
3	~~Sue~~ ~~Bob~~ Sue
4	Cal
5	Eve
6	~~Ted~~ ~~Sue~~ Sal
7	
8	Ted
9	Sal

5. We have discussed four uses of hash functions.

   (a) Perfect hashing.

   (b) Hash table implementation of search structures.

   (c) Cryptography.

   (d) Security, such as verification of passwords or other data.

   If you search for: "properties of a good hash function" you will get different lists. The list of desired properties depends on the use of the hash function. Write four such lists, one for each of the above uses. Some properties will be on each list, others on only some of the lists.

   Here are the properties.

   (i) The function must be deterministic, that is, each item h(x) is computed it returns the same value.

   (ii) The function must be fast to compute.

   (iii) Data which differ at just one bit must have very different hash values.

   (iv) There can be no collisions between two data items.

   (v) Collisions could be possible, but are impossible in practice. (That is, astronomically unlikely.)

   (vi) The function should be impossible in practice to invert, unless you have the right key, in which case the inverse function can be computed quickly.

   (vii) The function should be impossible in practice to invert, no matter what key you might have.

   (viii) The function should not be *too* fast to compute, since otherwise it could yield to a brute force attack.

   For each of the four applications of hash functions, list the properties which are needed for that application. Do not list properties which are not needed for that application.

   - List the needed properties of a perfect hash function.
     (i) (ii) (iv)

- List the needed properties of a hash function used for a search structure.

  (i) (ii) (iii)
- List the needed properties of a cryptographic hash function.

  (i) (ii) (iii) (iv) (vi) (viii)
- List the needed properties of a verification hash function.

  (i) (ii) (iii) (v) (viii)

6. Suppose you wish to store the values $\binom{n}{k}$ for all $n \leq N$ for some constant N. Recall that $0 \leq k \leq n$. These values form a triangular array, shown below for $N = 6$. To save space, you will store the values in an 1-dimensional array $A[M]$, where M is the size of the triangle. For example, $M = 21$ if $N = 6$. Write a function to fetch values. Here is your function, with one missing formula.

```
int choose(int n, int k)
 {
  int index = n(n+1)/2 + k;
  // fill in the formula for index
  return A[index];
 }
```

   Here is Pascal's triangle for $n \leq 6$.

```
 1
 1   1
 1   2   1
 1   3   3   1
 1   4   6   4   1
 1   5  10  10   5   1
 1   6  15  20  15   6   1
```

7. A 3-dimensional $9 \times 7 \times 10$ rectangular array A is stored in main memory in column-major order, and its base address is 8192. Each item of A takes two words of main memory, that is, two addressed locations. Find the address of A[4][5][2].

   The offset is $(9 * 7 * 2 + 9 * 5 + 4) * 2 = 350$. The base address of A[4][5][2] is 8192+350 = 8542.

8. Walk through Dijkstra's algorithm for the single source minpath problem for the directed graph illustrated below. Instead of numbering the vertices 0 through 19, I have assigned them letters from A to T. The source vertex is S.

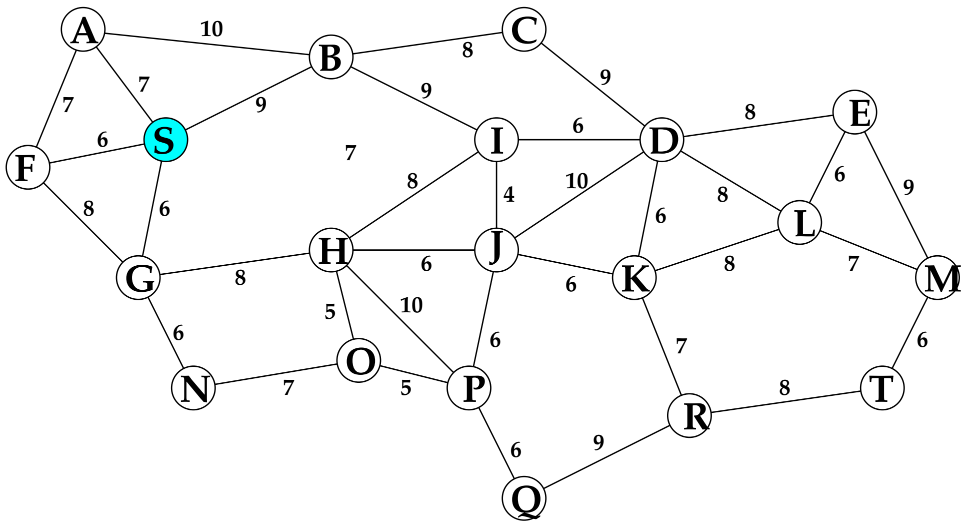
   After each iteration of the main loop, show

   1. The array dist, where dist[x] is the smallest length of any path found so far from S to x. (Initially, dist[x] = ∞ for most x.)

   2. The array back, where back[x] is the next-to-the last vertex on the path of smallest weight found so far from S to x.

   3 The contents of heap. Do not try to show the structure of the heap, simply list its members.

   Attach an extra sheet of paper if necessary.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist																			0	
back																				
heap	S																			

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9				6	6												0	
back	S	S				S	S													
heap	F	G	A	B																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9				6	6												0	
back	S	S				S	S													
heap	G	A	B																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9				6	6	14						12					0	
back	S	S				S	S	G						G						
heap	A	B	N	H																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9				6	6	14						12					0	
back	S	S				S	S	G						G						
heap	B	N	H																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17			6	6	14	18					12					0	
back	S	S	B			S	S	G	B					G						
heap	N	H	C	I																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17			6	6	14	18					12	19				0	
back	S	S	B			S	S	G	B					G	N					
heap	H	C	I	O																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17			6	6	14	18	20				12	19	24			0	
back	S	S	B			S	S	G	B	H				G	N	H				
heap	C	I	O	J	P															

dist	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	26		6	6	14	18	20				12	19	24			0	
back	S	S	B	C		S	S	G	B	H				G	N	H				
heap	I	O	J	P	D															

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24		6	6	14	18	20				12	19	24			0	
back	S	S	B	I		S	S	G	B	H				G	H	H				
heap	O	J	P	D																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24		6	6	14	18	20				12	19	24			0	
back	S	S	B	I		S	S	G	B	H				G	H	H				
heap	J	P	D																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24		6	6	14	18	20	26			12	19	24			0	
back	S	S	B	I		S	S	G	B	H	J			G	H	H				
heap	P	D	K																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24		6	6	14	18	20	26			12	19	24	30		0	
back	S	S	B	I		S	S	G	B	H	J			G	H	H	P			
heap	D	K	Q																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32		12	19	24	30		0	
back	S	S	B	I	D	S	S	G	B	H	J	D		G	H	H	P			
heap	K	Q	E	L																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32		12	19	24	30	33	0	
back	S	S	B	I	D	S	S	G	B	H	J	D		G	H	H	P	K		
heap	Q	E	L	R																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32		12	19	24	30	33	0	
back	S	S	B	I	D	S	S	G	B	H	J	D		G	H	H	P	K		
heap	E	L	R																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32	41	12	19	24	30	33	0	
back	S	S	B	I	D	S	S	G	B	H	J	D	E	G	H	H	P	K		
heap	L	R	M																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	0	
back	S	S	B	I	D	S	S	G	B	H	J	D	L	G	H	H	P	K		
heap	R	M																		

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	0	41
back	S	S	B	I	D	S	S	G	B	H	J	D	L	G	H	H	P	K		R
heap	M																			

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
dist	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	0	41
back	S	S	B	I	D	S	S	G	B	H	J	D	L	G	H	H	P	K		R
heap																				

Finished.