CSCE 313-200: Computer Systems

Homework #1 Part 3 (50 pts)

Due date: 2/20/25

1. Problem Description

In this part, you will multi-thread the previous version of the homework and analyze search results by answering a number of questions posed below.

1.1. Code (25 pts)

Your program must now accept four command-line arguments: planet, cave, number of threads to run, and the search method:

```
caveSearch.exe 6 44 300 BFS
```

First, make sure your program prints the initial CC exchange as in part 1, but omits the robot responses. Second, a dedicated stats thread must print in the following format every 2 seconds:

```
[2s] E 10.79K, U 23.83K, D 34.62K, 5362/sec, active 4894, run 5000
```

which shows that 2 seconds have elapsed, 10.79K rooms have been sent to robots for exploration (i.e., removed from *U*), 23.83K additional rooms are still pending in *U*, 34.62K rooms have been discovered, the current exploration rate is 5,362 rps (rooms/sec), there are 4,894 active threads with a room to explore, and a total of 5,000 threads are still running. Note that the first four values are *cumulative* (i.e., from the start time), while the exploration rate is computed since the last printout. The thread counts in the last two columns are *instantaneous*, i.e., taken when the printout is made.

When a search thread finds the exit, it must notify all others to stop searching and print its thread ID (in the range 0 to *N*–1), exit room number, how many rooms have been explored so far across all robots, and the distance along the discovered path:

```
Thread 9223: found exit AABF6237BFD5B685, steps 386416, distance 8
```

After all threads have shut down their robots and quit, the final printout summarizes the run. Note that the corresponding crawling rate is computed cumulatively since the beginning.

```
[final] E 413.90K, U 467.63K, D 881.53K, 12968/sec, active 0, run 0
Execution time: 34.66 seconds
```

It is not advisable to wake up threads if there is no work for them to do or constantly create/terminate them during the run (i.e., all *N* threads must be started at the beginning and kept going until the exit is found). Use the producer-consumer algorithm from class.

1.2. Report (25 pts)

It is recommended to allocate several days to analyze the data, run the experiments, and write about your results. Breakdown of points:

1. (5 pts) Disable the exit (i.e., search the entire graph) and experiment with BFS on planet 7, cave 55 by increasing the number of threads from 1 to 15,000 and document performance improvements arising from parallelization of the search. Specifically, plot the maximum stable search rate (e.g., after 120 seconds) vs. the number of threads and use curve-fitting to see how well this approximates a linear function. Using the slope of this curve, determine the average delay needed for a flybot to satisfy your request.
2. (5 pts) Use a *single* thread on planet 3, compute the *average* number of steps needed by each of the search methods (BFS, DFS, bFs, and A*) to find the exit in caves 40-49. Comment on the number of visited rooms by each method and the quality of the solution found (i.e., distance from the starting room). Does any one method win decisively?
3. (10 pts) Using 10K threads, plot a distribution of *shortest* distances (obtained using BFS) from the rover to each of the available rooms in cave 944 on planet $P = 6$. By eyeballing the figure, comment on whether this distribution can be approximated as Gaussian. To make results more meaningful, this should be plotted using a *normalized* histogram (also known as the *probability mass function*) where the y-axis contains the *percentage*, rather than the count, of rooms whose distance falls into a given bin on the x-axis. Figure 1(a) shows the result for planet 2; you will need a similar plot for $P = 6$. See http://en.wikipedia.org/wiki/Histogram and http://en.wikipedia.org/wiki/Probability_mass_function for more details.
4. (5 pts) Using 5K threads, study cave $C = 60$ on planets $P = 2$-7 and examine how the average *shortest* distance from the rover to all other rooms scales with planet size. It was observed that many random graphs asymptotically grow average distance as $\log_b(n)$, where n is the total number of nodes and b is a constant that depends on the average degree. Prove or disprove this conjecture and then experimentally determine b using curve fitting.

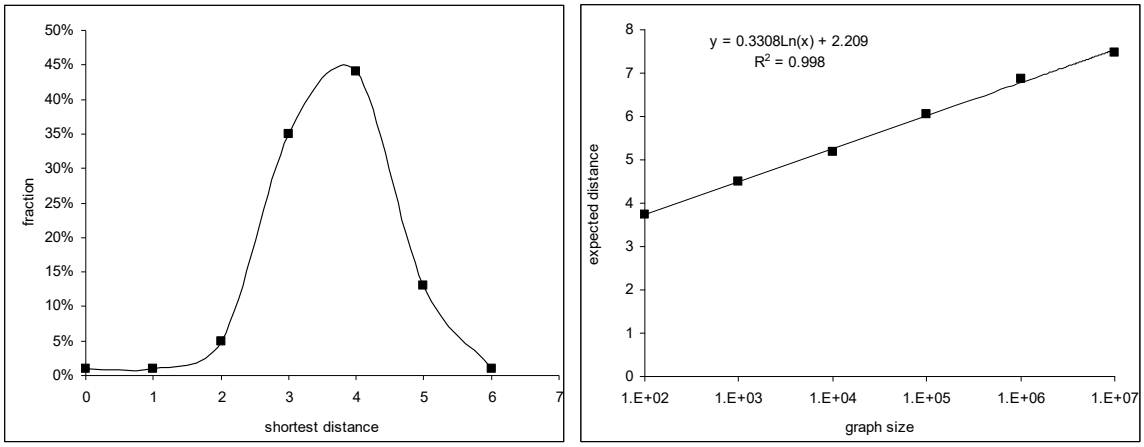

Figure 1. (a) Distribution of shortest distances for $P = 2$, $C = 44$. (b) Average distance for $C = 40$.

1.3. Extra Credit (20 pts)

Special caves on planets 6-7 are populated with a monster whose goal is to impede your escape. The main goal with this extra-credit assignment is to use overlapped I/O to overcome deadlocks and various message corruption to successfully navigate monster caves. All caves numbered 1,000 and higher have a monster in them.

The monster interferes in several ways – by eating a flybot, corrupting the response header, truncating the message, padding the room list, and crashing the process when you direct flybots to an invalid room. An eaten flybot results in either a timeout on `WriteFile`/`ReadFile` or an error in Windows APIs related to the pipe (e.g., `ReadFile`, `WriteFile`, `PeekNamedPipe`, `ConnectNamedPipe`, `WaitNamedPipe`, `WaitForSingleObject`, `GetOverlappedResult`). You must catch these conditions, place the room where the robot was killed back to the unexplored set *U*, and quit your thread that corresponds to this robot.

For jammed transmissions, the monster may corrupt the status code to become invalid (i.e., neither SUCCESS nor FAILURE) or truncate the message to a size smaller than the minimum valid length. It may also add bogus room numbers to the message in an attempt to confuse your search and ultimately cause the CC to crash when you attempt to navigate to an invalid room. However, when padding messages, the monster always produces a list in which the last `NodeTuple64` is incomplete (i.e., less than 12 bytes), which allows you to easily detect corrupted responses. For both jammed and padded messages, the correct course of action is to discard them and keep retrying the room in the same thread until success.

To help debug threads hanging in the CC, you can specify a command-line argument 'debug' to CC.exe, which will cause it to periodically print the status of its threads:

```
CC says: status = 1, msg = 'opened (planet 6, cave 1001) with 15000 robot(s)'
*** CC status: 1 (14963)  2 (32)  4 (5)
*** CC status: 1 (14479)  2 (85)  4 (436)
[2s] E 1.69K, U 15.38K, D 17.06K, 838/sec, active 876, run 879
*** CC status: 1 (14004)  2 (72)  4 (923)  7 (1)
*** CC status: 1 (13722)  2 (57)  4 (1218)  7 (2)  8 (1)
[4s] E 6.02K, U 52.15K, D 58.18K, 2146/sec, active 1498, run 1503
*** CC status: 1 (13315)  2 (116)  4 (1565)  5 (1)  7 (2)  8 (1)
*** CC status: 1 (12615)  2 (223)  4 (2155)  5 (3)  7 (1)  8 (3)
[6s] E 14.15K, U 107.82K, D 121.97K, 4038/sec, active 2961, run 2966
*** CC status: 1 (11594)  2 (292)  4 (3101)  5 (5)  7 (1)  8 (7)
*** CC status: 1 (9654)  2 (383)  4 (4946)  5 (2)  7 (1)  8 (14)
[8s] E 35.42K, U 183.32K, D 218.75K, 10565/sec, active 9540, run 9622
*** CC status: 2 (3154)  4 (11815)  7 (2)  8 (29)
*** CC status: 4 (14945)  7 (4)  8 (51)
[10s] E 85.47K, U 334.36K, D 419.83K, 24873/sec, active 14893, run 14953
*** CC status: 4 (14918)  7 (5)  8 (77)
```

The printouts show the number of threads (in parentheses) in each of the states 0 through 8. State 0 indicates the robot has not started yet, 1 is waiting for the pipe to be created in your thread, 2 waiting for the connect message, 4 expecting a MOVE command, 5 navigating to a room, 7 sending back the neighbors, and 8 shutting down after a DISCONNECT or consumption by the monster. Additional codes 3 and 6 indicate error conditions related to invalid initial commands and insufficient length of messages, but these should not arise if your program successfully completes non-monster caves.

For example, the last line of the above run shows that 14.9K threads are waiting for the MOVE command, 5 are about to send a reply, and 77 have been eaten by the monster. This debug information may become helpful when some of the robots never quit and hang your threads indefinitely. Knowing what they are doing should aid in ensuring a clean shutdown.

1.4. Caveats

It is crucial that you put robot pipe creation and initial connections into separate threads rather than attempt to run them from `main()`. Otherwise, you will wait a looooooong time for the initialization to complete in the main thread.

When benchmarking with a large number of threads, *always run your code in release mode*, where STL is 50 times faster than in debug mode and occupies 50% less memory. STL is also notoriously slow in releasing memory when compiled in debug mode, which sometimes creates an illusion of a deadlock when your code frees a large chunk of RAM just before terminating.

To avoid swapping to disk and showing low performance in your report, check that the total memory usage in Task Manager is well below your physical RAM size. You can notice that something is wrong when increasing the number of threads beyond some threshold (such as 2,500) leads to lower performance. Keep in mind that under Win32/x86, each process is limited to 2 GB of RAM, which caps the number of threads to about 5-6K; however, even this requires changing the default thread stack size from 1 MB to 64 KB (see Project Properties → Linker → System → Stack Reserve & Commit Size). Without this modification, the maximum number of Win32/x86 threads you can run is about 1,400. Therefore, it is recommended to always compile your code in x64.

1.5. Traces

On some older CPU architectures, there may be a noticeable delay in getting threads to start. Additionally, it is possible that that at any given time some of the robots are inactive (i.e., not exploring any rooms), which is normal:

```
CC says: status = 1, msg = 'opened (planet 6, cave 15) with 15000 robot(s)'
Starting search using BFS...
[2s] E 0.00K, U 0.00K, D 0.00K, 1/sec, active 1, run 675
[4s] E 3.65K, U 31.17K, D 34.82K, 1812/sec, active 1411, run 1412
[6s] E 10.90K, U 87.07K, D 97.97K, 3600/sec, active 2489, run 2498
[8s] E 22.02K, U 152.73K, D 174.75K, 5526/sec, active 3891, run 3896
[10s] E 39.08K, U 224.62K, D 263.70K, 8474/sec, active 5701, run 5706
[12s] E 64.28K, U 301.83K, D 366.11K, 12524/sec, active 8260, run 8270
[14s] E 98.49K, U 372.63K, D 471.12K, 17002/sec, active 11218, run 11249
[16s] E 143.53K, U 432.16K, D 575.68K, 22373/sec, active 14494, run 14527
[18s] E 193.05K, U 472.75K, D 665.80K, 24614/sec, active 14981, run 15000
[20s] E 242.88K, U 490.74K, D 733.62K, 24751/sec, active 14952, run 15000
[22s] E 292.81K, U 493.04K, D 785.85K, 24811/sec, active 14959, run 15000
[24s] E 342.52K, U 484.69K, D 827.21K, 24702/sec, active 14955, run 15000
Thread 9223: found exit AABF6237BFD5B685, steps 386416, distance 8
[27s] E 386.52K, U 479.06K, D 865.57K, 11603/sec, active 0, run 0
[final] E 386.52K, U 479.06K, D 865.57K, 13834/sec, active 0, run 0
Waiting for CC.exe to quit...
*** CC: main thread waiting for robots...
*** CC: all robots finished
*** CC: quitting, kernel time 0.14 sec, user time 0.18 sec
Execution time: 34.66 seconds
```

Another example are caves 900-999 that do not have an exit:

```
CC says: status = 1, msg = 'opened (planet 6, cave 950) with 9000 robot(s)'
Starting search using BFS...
[2s] E 4.79K, U 5.13K, D 9.92K, 2383/sec, active 4213, run 4230
[4s] E 31.01K, U 166.70K, D 197.72K, 13023/sec, active 8968, run 9000
[6s] E 60.96K, U 286.30K, D 347.26K, 14887/sec, active 8971, run 9000
[8s] E 90.92K, U 361.51K, D 452.43K, 14888/sec, active 8973, run 9000
...
[66s] E 955.71K, U 42.20K, D 997.91K, 14884/sec, active 8965, run 9000
```

```
[68s] E 985.74K, U 13.56K, D 999.30K, 14909/sec, active 8967, run 9000
[70s] E 999.99K, U 0.00K, D 999.99K, 7068/sec, active 220, run 9000
Thread 4837: reached empty queue!!
[final] E 1000.00K, U 0.00K, D 1000.00K, 13975/sec, active 0, run 0
Waiting for CC.exe to quit...
*** CC: main thread waiting for robots...
*** CC: all robots finished
*** CC: quitting, kernel time 0.12 sec, user time 0.22 sec
Execution time: 75.22 seconds
```

1.6. Monster Caves

The most interesting aspect here is that your code will lose threads at an average rate of one robot per 1,000 explored rooms. You will thus need at least 10K threads to finish planet 7, but 15-20K is recommended to keep the speed reasonable towards the end. You should also make sure that your program can handle the robot being killed *inside* the exit room as shown below.

```
Starting search using BFS...
CC says: status = 1, msg = 'opened (planet 6, cave 1001) with 15000 robot(s)'
[2s] E 1.43K, U 5.88K, D 7.31K, 710/sec, active 926, run 930
[4s] E 5.98K, U 43.30K, D 49.28K, 2256/sec, active 1669, run 1670
[6s] E 13.84K, U 102.92K, D 116.75K, 3906/sec, active 2677, run 2686
[8s] E 28.55K, U 165.92K, D 194.47K, 7313/sec, active 6652, run 6713
*** Monster: killing robot in exit room! ------------------------
[10s] E 92.36K, U 347.91K, D 440.26K, 24122/sec, active 14943, run 14949
*** Monster: killing robot in exit room! ------------------------
[12s] E 141.61K, U 428.57K, D 570.18K, 24466/sec, active 14880, run 14913
*** Monster: killing robot in exit room! ------------------------
[14s] E 190.63K, U 471.13K, D 661.76K, 24365/sec, active 14826, run 14870
[16s] E 239.81K, U 490.40K, D 730.21K, 24428/sec, active 14771, run 14814
*** Monster: killing robot in exit room! ------------------------
[18s] E 288.56K, U 493.34K, D 781.90K, 24227/sec, active 14708, run 14760
[20s] E 337.05K, U 485.99K, D 823.03K, 24094/sec, active 14653, run 14706
*** Monster: killing robot in exit room! ------------------------
[22s] E 385.22K, U 470.97K, D 856.20K, 23933/sec, active 14618, run 14656
*** Monster: killing robot in exit room! ------------------------
[24s] E 433.62K, U 449.66K, D 883.27K, 24048/sec, active 14565, run 14611
[26s] E 481.68K, U 423.61K, D 905.28K, 23870/sec, active 14527, run 14565
*** Monster: killing robot in exit room! ------------------------
[28s] E 529.82K, U 393.83K, D 923.65K, 23922/sec, active 14487, run 14518
Thread 5284: found exit 6DBA04BEDEC9B2A0, steps 567433, distance 6
[32s] E 567.51K, U 372.71K, D 940.22K, 10501/sec, active 8, run 8
[34s] E 567.51K, U 372.71K, D 940.22K, 0/sec, active 0, run 0
[final] E 567.51K, U 372.71K, D 940.22K, 16528/sec, active 0, run 0
```

313 Homework 1 Grade Sheet (Part 3)

Name: _____

Function	Points	Break down	Item	Points
Basic code structure	5	2	WaitForMultipleObjects(quit, semaQ)	
		1	Semaphore release	
		2	Use of mutex to protect U and D	
Speed & RAM	4	2	8000 rps with 5000 threads	
		2	Reasonable RAM utilization	
Functionality	14	3	BFS works with 5000 threads on P=6	
		3	DFS works with 5000 threads on P=6	
		3	bFS works with 5000 threads on P=6	
		3	A* works with 5000 threads on P=6	
		1	All threads quit on exit	
		1	All threads quit in caves 900-999	
Printouts	2	1	Statistics every 2 sec	
		1	Exit, distance, # of steps	
Misc*				

Extra Credit	20	10	Handle deadlock with overlapped I/O	
		10	Handle corrupted responses	

Report	25	5	Multi-core scalability analysis	
		5	Search method comparison	
		10	Distribution of distance in one planet	
		5	Average distance across planets	

Total points: _____