1 Exercise 1

1. a. A is possible to be the adjacency matrix of the graph. $D^{(2)}$ and $P^{(2)}$ is shown as follows.

$$D^{(2)} = \begin{pmatrix} 0 & \infty & \infty & \infty \\ 1 & 0 & \infty & 2 \\ \mathbf{3} & 2 & 0 & \mathbf{4} \\ -4 & \infty & \infty & 0 \end{pmatrix} P^{(2)} = \begin{pmatrix} Nil & Nil & Nil & Nil \\ 2 & Nil & Nil & 2 \\ \mathbf{2} & 3 & Nil & \mathbf{2} \\ 4 & Nil & Nil & Nil \end{pmatrix}$$

2
2.1 b. 20000/5=4000, 20000 disk pages and 5 main memory pages. this gives 20000/5=4000 runs.
2.2 b. We can afford up to 4-way as we have 5 main memory pages.
2.3 6. ceil($\log_{5-1}(4000)$) = 6 passes
2.4 28×10^4. 20000 disk pages and 5 main memory pages. this gives 20000/5=4000 runs. Second phase: $\lceil \log_4 4000 \rceil$=6 iterations. In total, (6+1)=7 times of read and write of the file. So $7 \times 2 \times 20000 = 28 \times 10^4$.

3 for AC
3.1. 4. slackness = parallelism / # of computation units. so 2*2=4
3.2 No, because the slackness is less than 1.

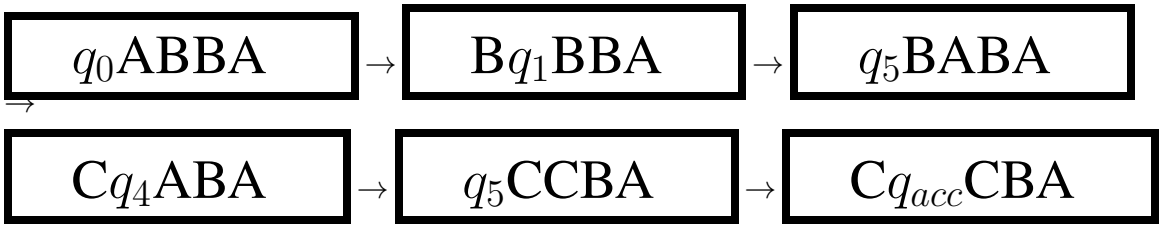
4
4.1 *PUSH*: 1, *POP*: 1, *COPY*: k.

4.2 Charge 2 for each *PUSH* and *POP* operation and 0 for each *COPY*. When we call *PUSH*, we use 1 to pay for the operation, and we store the other 1 on the item pushed. When we call *POP*, we again use 1 to pay for the operation, and we store the other 1 in the stack itself. Because the stack size never exceeds k, the actual cost of a *COPY* operation is at most k, which is paid by the k found in the items in the stack and the stack itself. Since there are k *PUSH* and *POP* operations between two consecutive *COPY* operations, there are k of credit stored, either on individual items (from *PUSH* operations) or in the stack itself (from *POP* operations) by the time a *COPY* occurs. Since the amortized cost of each operation is $\Theta(1)$ and the amount of credit never goes negative, the total cost of n operations is $\Theta(n)$.

Exercise 2, The C-Part, 50 points in total

1. Turing Machine.

1.1(*5 points*)



1.2 (*3 points*)

States $\{q_0, q_1, q_2, q_3, q_4, q_5, q_{acc}, q_{reject}\}$.

Input Alphabet {A,B} **Note: C can be optionally in the input alphabet as well.**.

Tape Alphabet {A,B,C,␣}.

1.3 (*2 points*) Consider that we run a TM M that has 10 states on an input w of length $|w| = 100$. If M accepts w, what (if any) is the maximum number of steps M can take before halting? Justify your answer.

Finite (as it eventually goes into q_{acc}), but arbitrarily large. While the input only has length w, the machine can write arbitrarily many cells and go right and left before stopping in q_{acc}.

Note, if the tape alphabet would be binary the answer could be the "Busy Beaver" number, but this has not been covered in the course, and here the alphabet is not restricted in any way.

2.1 (*4 points*) Let M be a TM.

a) If $L(M) \in P$ then M is guaranteed to run in polynomial time on the size of the input. ☐ **a)** Yes. ☒ **b)** No.

b) If M is guaranteed to run in polynomial time on the size of the input then $L(M) \in P$ ☒ **a)** Yes. ☐ **b)** No.

c) If $L(M) \in P$ then L(M) is decidable ☒ **a)** Yes. ☐ **b)** No.

d) If $L(M) \notin P$ then L(M) is undecidable ☐ **a)** Yes. ☒ **b)** No.

2.2 (12 points) Answer the following questions according to current knowledge (that is we do NOT assume that P = NP or P $\neq$ NP)

Let M be a TM that is a decider and runs in $O(n^{10})$. Mark all the classes to which $L(M)$ **may** belong.

☒ P ☒ NP ☒ NP-complete ☒ EXPTime

Let M be a TM that is a decider and runs in $O(n^{10})$. Mark all the classes to which $L(M)$ **must** belong.

☒ P ☒ NP ☐ NP-complete ☒ EXPTime

Let M be a TM that is a decider and runs in $O(10^n)$. Mark all the classes to which $L(M)$ **may** belong.

☒ P ☒ NP ☒ NP-complete ☒ EXPTime

Let M be a TM that is a decider and runs in $O(10^n)$. Mark all the classes to which $L(M)$ **must** belong.

☐ P ☐ NP ☐ NP-complete ☒ EXPTime

2.3 (4 points) Assume now that P $\neq$ NP, which of your answers would be different (if any). Justify your answer, explaining why the answer is different depending on whether we assume P $\neq$ NP or not.

The difference would be that in the first case (M runs in $O(n^{10})$), it may not belong to NP-complete, because in that case P=NP (as it suffices for a single NP-complete problem to be solvable in polynomial time).

3.1 (4 points) Assume that a language A is mapping reducible to language B. Which of the following claims are necessarily true?

a) If A is decidable then B is decidable too. ☐ **a)** Yes. ☒ **b)** No.

b) If A is undecidable then B is undecidable too. ☒ **a)** Yes. ☐ **b)** No.

3.2 (16 points) Consider the following languages. Mark **all** options that apply for each language. Justify your answer by giving a proof sketch (1) stating the proof technique you would use, and (2) providing the pseudocode of any TM that you'd use in that proof. Note: if you do not mark an option, your proof should explain why it is not decidable/recognizable/co-recognizable.

- $L_1 = \{\langle M, w, n\rangle \mid M \text{ is a TM that accepts } w \text{ on less than } n \text{ steps }\}$

  ☒ Decidable ☒ Recognizable ☒ Co-recognizable

  L_1 is decidable by a TM that does the following:

  Simulate M on w for n steps, and accept if M accepts, otherwise reject.

  This is a decider because it always terminates in a finite number of steps.

- $L_2 = \{\langle M, w, n\rangle \mid M \text{ is a TM that accepts } w \text{ on more than } n \text{ steps }\}$

  ☐ Decidable ☒ Recognizable ☐ Co-recognizable

  L_2 is recognizable but undecidable. It is recognized by a TM that does the following: Simulate M on w, counting the number of steps. If it accepts, and it has taken more than n steps accept.

  This is not a decider because M may never halt.

  To show that it is undecidable (so not co-recognizable), we use mapping reduction from A_{TM}:

  On input $\langle M, w\rangle$, write $\langle M, w, 0\rangle$.

  Clearly, M accepts w if and only if it accepts it after 0 steps.

- $L_3 = \{\langle n \rangle \mid n \text{ is a natural number}\}$

  [X] Decidable [X] Recognizable [X] Co-recognizable

  This is easily decidable. Just go over the input and check that all cells in the input tape correspond to digits and there are no comma or other symbols.

- $L_4 = \{\langle M_1, M_2, w \rangle \mid M_1 \text{ accepts } w \text{ or } M_2 \text{ rejects } w\}$

  [ ] Decidable [X] Recognizable [ ] Co-recognizable

  This is recognizable and undecidable.

  We show that it is undecidable by mapping reduction from $HALT$: $HALT \preceq_m L_4$.

  On input $\langle M, w \rangle$: write $\langle M, M, w \rangle$.

  Clearly, M halts on w if and only if M accepts w or M rejects w.

  We show that it is recognizable by providing a multitape TM, M_4

  On input $\langle M_1, M_2, w \rangle$:

  1. Copy w to second tape.
  2. Simulate M_1 in the first tape, M_2 in the second tape (simultaneously).
  3. If M_1 accepts or M_2 rejects then accept.
  4. Go to 2.

  Clearly, M_4 accepts if and only if M_1 accepts or M_2 rejects (even if the other never halts), so $L(M_4) = L_4$.
