This is a "core dump" of potential questions for the final exam. This should give you a good idea of the *types* of questions that we will ask on the exam. In particular, there will be a series of True/False or short-answer questions—but the actual exam questions may or may not appear in this handout. This list intentionally includes a few questions that are too long or difficult for exam conditions; these are indicated with a *star.

Don't forget to review the study problems for Midterms 1 and 2; the final exam is cumulative!

Solving every problem in this handout is *not* the best way to study for the exam. Memorizing the solutions to every problem in this handout is the *absolute worst* way to study for the exam.

What we recommend instead is to work on a *sample* of the problems. Choose one or two problems at random from each section and try to solve them from scratch under exam conditions—by yourself, in a quiet room, with a 30-minute timer, *without* your notes, *without* the internet, and if possible, even without your cheat sheet. If you're comfortable solving a few problems in a particular section, you're probably ready for that type of problem on the exam. Move on to the next section.

Discussing problems with other people (in your study groups, in the review sessions, in office hours, or on Ed/Discord) and/or looking up old solutions can be *extremely* helpful, but ***only after*** you have (1) made a good-faith effort to solve the problem on your own, and (2) you have either a candidate solution or some idea about where you're getting stuck.

If you find yourself getting stuck on a particular type of problem, try to figure out *why* you're stuck. Do you understand the problem statement? Are you stuck on choosing the right high-level approach? Are you stuck on the technical details? Or are you struggling to express your ideas clearly? (We *strongly* recommend writing solutions that follow the homework grading rubrics bullet-by-bullet.)

Similarly, if feedback from other people suggests that your solutions to a particular type of problem are incorrect or incomplete, try to figure out what you missed. For NP-hardness proofs: Are you choosing a good problem to reduce from? Are you reducing in the correct direction? Are you designing your reduction with both good instances and bad instances in mind? You're not trying *solve* the problem, are you? For undecidability proofs: Does the problem have the right structure to apply Rice's theorem? If you are arguing by reduction, are you reducing in the correct direction? You're not using *pronouns*, are you?

Remember that your goal is *not* merely to "understand" the solution to any particular problem, but to become more comfortable with solving a certain *type* of problem on your own. **"Understanding" is a trap; aim for mastery.** If you can identify specific steps that you find problematic, read more *about those steps*, focus your practice *on those steps*, and try to find helpful information *about those steps* to write on your cheat sheet. Then work on the next problem!

True or False?

For each statement below, write "YES" or "True" if the statement is ***always*** true and "NO" or "False" otherwise, and give a brief (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, write "NO" or "False". For example:

- $x + y = 5$
  NO — Suppose $x = 3$ and $y = 4$.
- 3SAT can be solved in polynomial time.
  NO — 3SAT is NP-hard.
- If P = NP then Jeff is the Queen of England.
  YES — The hypothesis is false, so the implication is true.

---

1 Which of the following are clear English specifications of a recursive function that could possibly be used to compute the edit distance between two strings $A[1..n]$ and $B[1..n]$?

1.A. $Edit(i, j)$ is the answer for i and j.

1.B. $Edit(i, j)$ is the edit distance between $A[i]$ and $B[j]$.

1.C.
$$Edit[i,j] = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ Edit[i-1, j-1] & \text{if } A[i] = B[j] \\ \max\{1 + Edit[i, j-1],\ 1 + Edit[i-1, j],\ 1 + Edit[i-1, j-1]\} & \text{otherwise} \end{cases}$$

1.D. $Edit[1..n, 1..n]$ stores the edit distances for all prefixes.

1.E. $Edit(i, j)$ is the edit distance between $A[i..n]$ and $B[j..n]$.

1.F. $Edit[i, j]$ is the value stored at row i and column j of the table.

1.G. $Edit(i, j)$ is the edit distance between the last i characters of A and the last j characters of B.

1.H. $Edit(i, j)$ is the edit distance when i and j are the current characters in A and B.

1.I. Iterate through both strings and update $Edit[\cdot, \cdot]$ at each character.

1.J. $Edit(i, j, k, l)$ is the edit distance between substrings $A[i..j]$ and $B[k..l]$.

1.K. *[I don't need an English description; my pseudocode is clear enough!]*

---

2 Which of the following statements is true for ***every*** directed graph $G = (V, E)$?

2.A. $E \neq \varnothing$.

2.B. Given the graph G as input, Floyd-Warshall runs in $O(E^3)$ time.

2.C. If G has at least one source and at least one sink, then G is a dag.

2.D. We can compute a spanning tree of G using whatever-first search.

2.E. If the edges of G are weighted, we can compute the shortest path from any node s to any node t in $O(E \log V)$ time using Dijkstra's algorithm.

---

3 Which of the following statements are true for ***every*** language $L \subseteq \{0, 1\}^*$?

3.A. L is non-empty.

3.B. L is infinite.

3.C. L contains the empty string ε.

3.D. L^* is infinite.

3.E. L^* is regular.

3.F. L is accepted by some DFA if and only if L is accepted by some NFA.

3.G. L is described by some regular expression if and only if L is rejected by some NFA.

3.H. L is accepted by some DFA with 42 states if and only if L is accepted by some NFA with 42 states.

3.I. If L is decidable, then L is infinite.

3.J. If L is not decidable, then L is infinite.

3.K. If L is not regular, then L is undecidable.

3.L. If L has an infinite fooling set, then L is undecidable.

3.M. If L has a finite fooling set, then L is decidable.

3.N. If L is the union of two regular languages, then its complement $\overline{L}$ is regular.

3.O. If L is the union of two regular languages, then its complement $\overline{L}$ is context-free.

3.P. If L is the union of two decidable languages, then L is decidable.

3.Q. If L is the union of two undecidable languages, then L is undecidable.

3.R. If $L \notin \mathrm{P}$, then L is not regular.

3.S. L is decidable if and only if its complement $\overline{L}$ is undecidable.

3.T. Both L and its complement $\overline{L}$ are decidable.

---

4 Which of the following statements are true for ***at least one*** language $L \subseteq \{0,1\}^*$?

4.A. L is non-empty.

4.B. L is infinite.

4.C. L contains the empty string ε.

4.D. L^* is finite.

4.E. L^* is not regular.

4.F. L is not regular but L^* is regular.

4.G. L is finite and L is undecidable.

4.H. L is decidable but L^* is not decidable.

4.I. L is not decidable but L^* is decidable.

4.J. L is the union of two decidable languages, but L is not decidable.

4.K. L is the union of two undecidable languages, but L is decidable.

4.L. L is accepted by an NFA with 374 states, but L is not accepted by a DFA with 374 states.

4.M. L is accepted by an DFA with 374 states, but L is not accepted by a NFA with 374 states.

4.N. L is regular and $L \notin \mathrm{P}$.

4.O. There is a Turing machine that accepts L.

4.P. There is an algorithm to decide whether an arbitrary given Turing machine accepts L.

---

5 Which of the following languages over the alphabet $\{0,1\}$ are ***regular***?

5.A. $\{\mathtt{0}^m\mathtt{1}^n \mid m \geq 0 \text{ and } n \geq 0\}$

5.B. All strings with the same number of 0s and 1s

5.C. Binary representations of all positive integers divisible by 17

5.D. Binary representations of all prime numbers less than 10^{100}

5.E. $\{ww \mid w \text{ is a palindrome}\}$

5.F. $\{wxw \mid w \text{ is a palindrome and } x \in \{\mathtt{0},\mathtt{1}\}^*\}$

5.G. $\{\langle M \rangle \mid M \text{ accepts a regular language}\}$

5.H. $\{\langle M \rangle \mid M \text{ accepts a finite number of non-palindromes}\}$

---

6 Which of the following languages/decision problems are ***decidable***?

6.A. $\varnothing$

6.B. $\{\mathtt{0}^n\mathtt{1}^{2n}\mathtt{0}^n\mathtt{1}^{2n} \mid n \geq 0\}$

6.C. $\{ww \mid w \text{ is a palindrome}\}$

6.D. $\{\langle M \rangle \mid M \text{ accepts } \langle M \rangle \bullet \langle M \rangle\}$

6.E. $\{\langle M \rangle \mid M \text{ accepts a finite number of non-palindromes}\}$

6.F. $\{\langle M \rangle \bullet w \mid M \text{ accepts } ww\}$

6.G. $\left\{\langle M \rangle \bullet w \;\middle|\; M \text{ accepts } ww \text{ after at most } |w|^2 \text{ steps}\right\}$

6.H. Given an NFA N, is the language $L(N)$ infinite?

6.I. SAT

6.J. Given an undirected graph G, does G contain a Hamiltonian cycle?

6.K. Given encodings of two Turing machines M and M', is there a string w that is accepted by both M and M'?

---

7 Recall the halting language $\text{HALT} = \{\langle M \rangle \bullet w \mid M \text{ halts on input } w\}$. Which of the following statements about its complement $\overline{\text{HALT}} = \Sigma^* \setminus \text{HALT}$ are true?

7.A. $\overline{\text{HALT}}$ is empty.

7.B. $\overline{\text{HALT}}$ is regular.

7.C. $\overline{\text{HALT}}$ is infinite.

7.D. $\overline{\text{HALT}}$ is in NP.

7.E. $\overline{\text{HALT}}$ is decidable.

---

8 Suppose some language $A \in \{\mathtt{0},\mathtt{1}\}^*$ reduces to another language $B \in \{\mathtt{0},\mathtt{1}\}^*$. Which of the following statements ***must*** be true?

8.A. A Turing machine that recognizes A can be used to construct a Turing machine that recognizes B.

8.B. A is decidable.

8.C. If B is decidable then A is decidable.

8.D. If A is decidable then B is decidable.

8.E. If B is NP-hard then A is NP-hard.

8.F. If A has no polynomial-time algorithm then neither does B.

---

9 Suppose there is a ***polynomial-time*** reduction from problem A to problem B. Which of the following statements ***must*** be true?

9.A. Problem B is NP-hard.

9.B. A polynomial-time algorithm for B can be used to solve A in polynomial time.

9.C. If B has no polynomial-time algorithm then neither does A.

9.D. If A is NP-hard and B has a polynomial-time algorithm then P = NP.

9.E. If B is NP-hard then A is NP-hard.

9.F. If B is undecidable then A is undecidable.

---

10 Consider the following pair of languages:

- HAMPATH $:= \{G \mid G \text{ is an undirected graph with a Hamiltonian path}\}$
- CONNECTED $:= \{G \mid G \text{ is a connected undirected graph}\}$

(For concreteness, assume that in both of these languages, graphs are represented by their adjacency matrices.) Which of the following ***must*** be true, assuming P≠NP?

10.A. CONNECTED $\in$ NP

10.B. HAMPATH $\in$ NP

10.C. HAMPATH is decidable.

10.D. There is no polynomial-time reduction from HAMPATH to CONNECTED.

10.E. There is no polynomial-time reduction from CONNECTED to HAMPATH.

---

11 Consider the following pair of languages:

- DIRHAMPATH $:= \{G \mid G \text{ is a directed graph with a Hamiltonian path}\}$
- ACYCLIC $:= \{G \mid G \text{ is a directed acyclic graph}\}$

(For concreteness, assume that in both of these languages, graphs are represented by their adjacency matrices.) Which of the following ***must*** be true, assuming P≠NP?

11.A. ACYCLIC $\in$ NP

11.B. ACYCLIC $\cap$ DIRHAMPATH $\in$ P

11.C. DIRHAMPATH is decidable.

11.D. There is a polynomial-time reduction from DIRHAMPATH to ACYCLIC.

11.E. There is a polynomial-time reduction from ACYCLIC to DIRHAMPATH.

---

12 Consider the following pair of languages:

- 3COLOR $:= \{G \mid G \text{ is a 3-colorable undirected graph}\}$
- TREE $:= \{G \mid G \text{ is a connected acyclic undirected graph}\}$

(For concreteness, assume that in both of these languages, graphs are represented by adjacency matrices.) Which of the following ***must*** be true, assuming P≠NP?

12.A. TREE is NP-hard.

12.B. $\textsc{Tree} \cap \textsc{3Color} \in \mathrm{P}$

12.C. $\textsc{3Color}$ is undecidable.

12.D. There is a polynomial-time reduction from $\textsc{3Color}$ to $\textsc{Tree}$.

12.E. There is a polynomial-time reduction from $\textsc{Tree}$ to $\textsc{3Color}$.

---

13 Suppose we want to prove that the following language is undecidable.

$$\textsc{AlwaysHalts} := \{\langle M \rangle \mid M \text{ halts on every input string}\}$$

Rocket J. Squirrel suggests a reduction from the standard halting language

$$\textsc{Halt} := \{(\langle M \rangle, w) \mid M \text{ halts on inputs } w\}.$$

Specifically, given a Turing machine $\textsc{DecideAlwaysHalts}$ that decides $\textsc{AlwaysHalts}$, Rocky claims that the following Turing machine $\textsc{DecideHalt}$ decides $\textsc{Halt}$.

```
DecideHalt(⟨M⟩, w):
Write code for the following algorithm:
  Bullwinkle(x):
    if M accepts w
      reject
    if M rejects w
      accept
return DecideAlwaysHalts(⟨Bullwinkle⟩)
```

Which of the following statements is true ***for all*** inputs $\langle M \rangle \# w$?

13.A. If M accepts w, then M' halts on every input string.

13.B. If M rejects w, then M' halts on every input string.

13.C. If M diverges on w, then M' halts on every input string.

13.D. If M accepts w, then $\textsc{DecideAlwaysHalts}$ accepts $\langle \textsc{Bullwinkle} \rangle$.

13.E. If M rejects w, then $\textsc{DecideHalt}$ rejects $(\langle M \rangle, w)$.

13.F. If M diverges on w, then $\textsc{DecideAlwaysHalts}$ diverges on $\langle \textsc{Bullwinkle} \rangle$.

13.G. $\textsc{DecideHalt}$ decides $\textsc{Halt}$. (That is, Rocky's reduction is correct.)

---

14 Suppose we want to prove that the following language is undecidable.

$$\textsc{Muggle} := \{\langle M \rangle \mid M \text{ accepts } \texttt{SCIENCE} \text{ but rejects } \texttt{MAGIC}\}$$

Professor Potter, your instructor in Defense Against Models of Computation and Other Dark Arts, suggests a reduction from the standard halting language

$$\textsc{Halt} := \{(\langle M \rangle, w) \mid M \text{ halts on inputs } w\}.$$

Specifically, suppose there is a Turing machine $\textsc{DetectoMuggletum}$ that decides $\textsc{Muggle}$. Professor Potter claims that the following algorithm decides $\textsc{Halt}$.

```
DecideHalt(⟨M⟩, w):
Write code for the following algorithm:
  RubberDuck(x):
    run M on input w
    if x = MAGIC
        return False
    else
        return True
return DetectoMuggletum(⟨RubberDuck⟩)
```

Which of the following statements ***must be*** true ***for all*** inputs $\langle M \rangle \# w$?

14.A. If M accepts w, then RubberDuck accepts SCIENCE.

14.B. If M accepts w, then RubberDuck accepts CHOCOLATE.

14.C. If M rejects w, then RubberDuck rejects MAGIC.

14.D. If M rejects w, then RubberDuck halts on every input string.

14.E. If M diverges on w, then RubberDuck rejects every input string.

14.F. If M accepts w, then DetectoMuggletum accepts $\langle$RubberDuck$\rangle$.

14.G. If M rejects w, then DecideHalt rejects $(\langle M \rangle, w)$.

14.H. If M diverges on w, then DecideHalt rejects $(\langle M \rangle, w)$.

14.I. DecideHalt decides the language Halt. (That is, Professor Potter's reduction is actually correct.)

14.J. DecideHalt actually runs (or simulates) RubberDuck.

14.K. Muggle is decidable.

---

15 Suppose we want to prove that the following language is undecidable.

$$\text{Marvin} := \left\{ \langle M \rangle \;\middle|\; M \text{ rejects an infinite number of strings} \right\}$$

Professor Beeblebrox, your instructor in Infinitely Improbable Galactic Presidencies, suggests a reduction from the standard halting language

$$\text{Halt} := \left\{ (\langle M \rangle, w) \;\middle|\; M \text{ halts on inputs } w \right\}.$$

Specifically, suppose there is a program ParanoidAndroid that decides Marvin. Professor Beeblebrox claims that the following algorithm decides Halt.

```
DecideHalt(⟨M⟩, w):
Write code for the following algorithm:
  HeartOfGold(x):
    run M on input w
    if x = VOGONPOETRY
        return False
    else
        return True
return ParanoidAndroid(⟨HeartOfGold⟩)
```

Which of the following statements is true for all inputs $(\langle M \rangle, w)$?

15.A. If M accepts w, then HeartOfGold accepts VOGONPOETRY.

15.B. If M accepts w, then HeartOfGold accepts IMPROBABILITY.

15.C. If M rejects w, then HeartOfGold rejects VOGONPOETRY.

15.D. If M rejects w, then HeartOfGold rejects IMPROBABILITY.

15.E. If M hangs on w, then HeartOfGold accepts VOGONPOETRY.

15.F. If M hangs on w, then HeartOfGold rejects IMPROBABILITY.

15.G. If M hangs on w, then HeartOfGold hangs on NEILYOUNG.

15.H. ParanoidAndroid accepts OKCOMPUTER.

15.I. ParanoidAndroid accepts $\langle$HeartOfGold$\rangle$.

15.J. ParanoidAndroid rejects $\langle$HeartOfGold$\rangle$.

15.K. ParanoidAndroid actually runs (or simulates running) HeartOfGold.

15.L. DecideHalt decides Halt; that is, Professor Beeblebrox's proof is correct.

NP-Hardness

16 A boolean formula is in *disjunctive normal form* (or *DNF*) if it consists of a *disjunction* (OR) or several *terms*, each of which is the conjunction (AND) of one or more literals. For example, the formula

$$(\overline{x} \wedge y \wedge \overline{z}) \vee (y \wedge z) \vee (x \wedge \overline{y} \wedge \overline{z})$$

is in disjunctive normal form. DNF-SAT asks, given a boolean formula in disjunctive normal form, whether that formula is satisfiable.

16.A. Describe a polynomial-time algorithm to solve DNF-SAT.

16.B. What is the error in the following argument that P=NP?

> *Suppose we are given a boolean formula in conjunctive normal form with at most three literals per clause, and we want to know if it is satisfiable. We can use the distributive law to construct an equivalent formula in disjunctive normal form. For example,*
>
> $$(x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y}) \iff (x \wedge \overline{y}) \vee (y \wedge \overline{x}) \vee (\overline{z} \wedge \overline{x}) \vee (\overline{z} \wedge \overline{y})$$
>
> *Now we can use the algorithm from part (a) to determine, in polynomial time, whether the resulting DNF formula is satisfiable. We have just solved* 3SAT *in polynomial time. Since* 3SAT *is NP-hard, we must conclude that P=NP!*

17 A ***relaxed 3-coloring*** of a graph G assigns each vertex of G one of three colors (for example, red, green, and blue), such that ***at most one*** edge in G has both endpoints the same color.

17.A. Give an example of a graph that has a relaxed 3-coloring, but does not have a proper 3-coloring (where every edge has endpoints of different colors).

17.B. ***Prove*** that it is NP-hard to determine whether a given graph has a relaxed 3-coloring.

18 An ***ultra-Hamiltonian tour*** in G is a closed walk W that visits every vertex of G exactly once, except for *at most one* vertex that W visits more than once.

18.A. Give an example of a graph that contains a ultra-Hamiltonian tour, but does not contain a Hamiltonian cycle (which visits every vertex exactly once).

18.B. ***Prove*** that it is NP-hard to determine whether a given graph contains a ultra-Hamiltonian tour.

19 An ***infra-Hamiltonian cycle*** in G is a closed walk W that visits every vertex of G exactly once, except for *at most one* vertex that W does not visit at all.

19.A. Give an example of a graph that contains a infra-Hamiltonian cycle, but does not contain a Hamiltonian cycle (which visits every vertex exactly once).

19.B. ***Prove*** that it is NP-hard to determine whether a given graph contains a infra-Hamiltonian cycle.

20 A ***quasi-satisfying assignment*** for a 3CNF boolean formula Φ is an assignment of truth values to the variables such that *at most one* clause in Φ does not contain a true literal. ***Prove*** that it is NP-hard to determine whether a given 3CNF boolean formula has a quasi-satisfying assignment.

21 A subset S of vertices in an undirected graph G is ***half-independent*** if each vertex in S is adjacent to *at most one* other vertex in S. Prove that finding the size of the largest half-independent set of vertices in a given undirected graph is NP-hard.

22 A subset S of vertices in an undirected graph G is ***sort-of-independent*** if if each vertex in S is adjacent to *at most 374* other vertices in S. Prove that finding the size of the largest sort-of-independent set of vertices in a given undirected graph is NP-hard.

23 A subset S of vertices in an undirected graph G is ***almost independent*** if at most 374 edges in G have both endpoints in S. Prove that finding the size of the largest almost-independent set of vertices in a given undirected graph is NP-hard.

24 Let G be an undirected graph with weighted edges. A ***heavy Hamiltonian cycle*** is a cycle C that passes through each vertex of G exactly once, such that the total weight of the edges in C is more than half of the total weight of all edges in G. Prove that deciding whether a graph has a heavy Hamiltonian cycle is NP-hard.

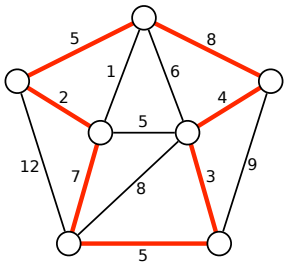

Figure 1: A heavy Hamiltonian cycle. The cycle has total weight 34; the graph has total weight 67.

25 **25.A.** A ***tonian path*** in a graph G is a path that goes through at least half of the vertices of G. Show that determining whether a graph has a tonian path is NP-hard.

25.B. A ***tonian cycle*** in a graph G is a cycle that goes through at least half of the vertices of G. Show that determining whether a graph has a tonian cycle is NP-hard. (**Hint:** Use part (a). Or not.)

26 Prove that the following variants of SAT is NP-hard. (**Hint:** Describe reductions from 3SAT.)

26.A. Given a boolean formula Φ in conjunctive normal form, where *each variable appears in at most three clauses*, determine whether Φ has a satisfying assignment. (**Hint:** First consider the variant where each variable appears in at most **five** clauses.)

26.B. Given a boolean formula Φ in conjunctive normal form *and given one satisfying assignment for* Φ, determine whether Φ has at least one other satisfying assignment.

27 Jerry Springer and Maury Povich have decided not to compete with each other over scheduling guests during the next talk-show season. There is only one set of Weird People who either host would consider having on their show. The hosts want to divide the Weird People into two disjoint subsets: those to appear on Jerry's show, and those to appear on Maury's show. (Neither wants to "recycle" a guest that appeared on the other's show.)

Both Jerry and Maury have preferences about which Weird People they are particularly interested in. For example, Jerry wants at least one guest who fits the description "was abducted by a flying saucer". Thus, on his list of preferences, he writes "w_1 or w_3 or w_{45}", since weird people numbered 1, 3, and 45 are the only ones who fit that description. Jerry has other preferences as well, so he lists those also. Similarly, Maury might like to include at least one guest who "really enjoys Rice's theorem". Each potential guest may fall into any number of different categories, such as the person who enjoys Rice's theorem more than their involuntary flying-saucer voyage.

Jerry and Maury each prepare a list reflecting all of their preferences. Each list contains a collection of statements of the form "(w_i or w_j or w_k)". Your task is to prove that it is NP-hard to find an assignment of weird guests to the two shows that satisfies all of Jerry's preferences and all of Maury's preferences.

27.A. The problem NoMixedClauses3Sat is the special case of 3Sat where the input formula cannot contain a clause with both a negated variable and a non-negated variable. Prove that NoMixedClauses3Sat is NP-hard. (**Hint:** Reduce from the standard 3Sat problem.)

27.B. Describe a polynomial-time reduction from NoMixedClauses3Sat to 3Sat.

28 The president of Sham-Poobanana University is planning An Unofficial St. Brigid's Day party for the university staff.[1] His staff has a hierarchical structure; that is, the supervisor relation forms a directed, acyclic graph, with the president as the only source, and with an edge from person i to person j in the graph if and only if person i is an immediate supervisor of person j. (Many staff members have multiple positions, and thus have several immediate supervisors.) In order to make the party fun for all guests, the president wants to ensure that if a person i attends, then none of i's immediate supervisors can attend.

By mining each staff member's email and social media accounts, Sham-Poobanana University Human Resources has determined a "party-hound" rating for each staff member, which is a non-negative real number reflecting how likely it is that the person will leave the party wearing a monkey suit and a lampshade.

Show that it is NP-hard to determine a guest-list that *maximizes* the sum of the party-hound ratings of all invited guests, subject to the supervisor constraint.

(**Hint:** This problem can be solved in polynomial time when the input graph is a tree!)

29 Prove that the following problem (which we call Match) is NP-hard. The input is a finite set S of strings, all of the same length n, over the alphabet $\{0, 1, 2\}$. The problem is to determine whether there is a string $w \in \{0, 1\}^n$ such that for every string $s \in S$, the strings s and w have the same symbol in at least one position.

For example, given the set $S = \{01220, 21110, 21120, 00211, 11101\}$, the correct output is True, because the string $w = 01001$ matches the first three strings of S in the second position, and matches the last two strings of S in the last position. On the other hand, given the set $S = \{00, 11, 01, 10\}$, the correct output is False.

(**Hint:** Describe a reduction from SAT (or 3SAT))

30 Consider the following solitaire game. The puzzle consists of an $n \times m$ grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions:

(1) Every row contains at least one stone.

(2) No column contains stones of both colors.

For some initial configurations of stones, reaching this goal is impossible; see the example below.

Prove that it is NP-hard to determine, given an initial configuration of red and blue stones, whether this puzzle can be solved.

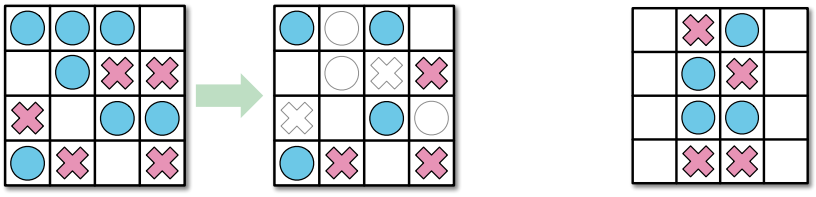

Figure 2: (Left) A solvable puzzle and one of its many solutions. (Right) An unsolvable puzzle.

[1] As I'm sure you already know, St. Brigid of Kildare is one of the patron saints of Ireland, chicken and dairy farmers, and academics.

31 To celebrate the end of the semester, Professor Jarling wants to treat himself to an ice-cream cone at the *Polynomial House of Flavors*. For a fixed price, he can build a cone with as many scoops as he'd like. Because he has good balance (and because we want this problem to work out), Prof. Jarling can balance any number of scoops on top of the cone without it tipping over. He plans to eat the ice cream one scoop at a time, from top to bottom, and doesn't want more than one scoop of any flavor.

However, he realizes that eating a scoop of bubblegum ice cream immediately after the scoop of potatoes-and-gravy ice cream would be unpalatable; these two flavors clearly should not be placed next to each other in the stack. He has other similar constraints; certain pairs of flavors cannot be adjacent in the stack.

He'd like to get as much ice cream as he can for the one fee by building the tallest cone possible that meets his flavor-incompatibility constraints. Prove that Prof. Jarling's problem is NP-hard.

32 Prove that the following problems are NP-hard.

32.A. Given an undirected graph G, does G contain a simple path that visits all but 17 vertices?

32.B. Given an undirected graph G, does G have a spanning tree in which every node has degree at most 23?

32.C. Given an undirected graph G, does G have a spanning tree with at most 42 leaves?

33 Prove that the following problems are NP-hard.

33.A. Given an undirected graph G, is it possible to color the vertices of G with three different colors, so that at most 31337 edges have both endpoints the same color?

33.B. Given an undirected graph G, is it possible to color the vertices of G with three different colors, so that each vertex has at most 8675309 neighbors with the same color?

34 At the end of every semester, Jeff needs to solve the following ExamDesign problem. He has a list of problems, and he knows for each problem which students will *really enjoy* that problem. He needs to choose a subset of problems for the exam such that for each student in the class, the exam includes at least one question that student will really enjoy. On the other hand, he does not want to spend the entire summer grading an exam with dozens of questions, so the exam must also contain as few questions as possible. Prove that the ExamDesign problem is NP-hard.

35 Which of the following results would resolve the P vs. NP question? Justify each answer with a short sentence or two.

35.A. The construction of a polynomial time algorithm for some problem in NP.

35.B. A polynomial-time reduction from 3Sat to the language $\{0^n1^n \mid n \geq 0\}$.

35.C. A polynomial-time reduction from $\{0^n1^n \mid n \geq 0\}$ to 3Sat.

35.D. A polynomial-time reduction from 3Color to MinVertexCover.

35.E. The construction of a nondeterministic Turing machine that cannot be simulated by any deterministic Turing machine with the same running time.

Greedy Algorithms

Remember that you will receive ***zero*** points for a greedy algorithm, even if it is perfectly correct, unless you also give a formal proof of correctness.

36 Recall the class scheduling problem described in lecture on Tuesday. We are given two arrays $S[1..n]$ and $F[1..n]$, where $S[i] < F[i]$ for each i, representing the start and finish times of n classes. Your goal is to find the largest number of classes you can take without ever taking two classes simultaneously. We showed in class that the following greedy algorithm constructs an optimal schedule:

> Choose the course that *ends first*, discard all conflicting classes, and recurse.

But this is not the only greedy strategy we could have tried. For each of the following alternative greedy algorithms, either prove that the algorithm always constructs an optimal schedule, or describe a small input example for which the algorithm does not produce an optimal schedule. Assume that all algorithms break ties arbitrarily (that is, in a manner that is completely out of your control).

(**Hint:** Exactly three of these greedy strategies actually work.)

36.A. Choose the course x that *ends last*, discard classes that conflict with x, and recurse.

36.B. Choose the course x that *starts first*, discard all classes that conflict with x, and recurse.

36.C. Choose the course x that *starts last*, discard all classes that conflict with x, and recurse.

36.D. Choose the course x with *shortest duration*, discard all classes that conflict with x, and recurse.

36.E. Choose a course x that *conflicts with the fewest other courses*, discard all classes that conflict with x, and recurse.

36.F. If no classes conflict, choose them all. Otherwise, discard the course with *longest duration* and recurse.

36.G. If no classes conflict, choose them all. Otherwise, discard a course that *conflicts with the most other courses* and recurse.

36.H. Let x be the class with the *earliest start time*, and let y be the class with the *second earliest start time*.

- If x and y are disjoint, choose x and recurse on everything but x.
- If x completely contains y, discard x and recurse.
- Otherwise, discard y and recurse.

36.I. If any course x completely contains another course, discard x and recurse. Otherwise, choose the course y that *ends last*, discard all classes that conflict with y, and recurse.

37 Binaria uses coins whose values are $1, 2, 4, \ldots, 2^k$, the first k powers of two, for some integer k. As in most countries, Binarian shopkeepers always make change using the following greedy algorithm:

```
MakeChange(N):
  if N = 0
      say "Thank you, come again!"
  else
      c ← largest coin value such that c ≤ N
      give the customer one c cent coin
      MakeChange(N − c)
```

For example, to make 37 cents in change, the shopkeeper would give the customer one 32 cent coin, one 4 cent coin, and one 1 cent coin, and then say "Thank you, come again!" (For purposes of this problem, assume that every shopkeeper has an unlimited supply of each type of coin.)

Prove that this greedy algorithm always uses the smallest possible number of coins. (**Hint:** Prove that the greedy algorithm uses at most one coin of each denomination.)

38 Let X be a set of n intervals on the real line. We say that a set P of points *stabs* X if every interval in X contains at least one point in P. Describe and analyze an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists of two arrays $L[1 \,..\, n]$ and $R[1 \,..\, n]$, representing the left and right endpoints of the intervals in X. If you use a greedy algorithm, don't forget to ***prove*** that it is correct.

Turing Machines and Undecidability

The only undecidability questions on this semester's final exam will be True/False or short-answer, but the following problems might still be useful to build intuition.

For each of the following languages, either ***sketch*** an algorithm to decide that language or ***prove*** that the language is undecidable, using a diagonalization argument, a reduction argument, closure properties, or some combination of the above. Recall that w^R denotes the reversal of string w.

39 $\varnothing$

40 $\{\texttt{0}^n\texttt{1}^n\texttt{2}^n \mid n \geq 0\}$

41 $\{A \in \{\texttt{0},\texttt{1}\}^{n \times n} \mid n \geq 0 \text{ and } A \text{ is the adjacency matrix of a dag with } n \text{ vertices}\}$

42 $\{A \in \{\texttt{0},\texttt{1}\}^{n \times n} \mid n \geq 0 \text{ and } A \text{ is the adjacency matrix of a 3-colorable graph with } n \text{ vertices}\}$

43 $\{\langle M \rangle \mid M \text{ accepts } \langle M \rangle^R\}$

44 $\{\langle M \rangle \mid M \text{ accepts } \langle M \rangle^R\} \cap \{\langle M \rangle \mid M \text{ rejects } \langle M \rangle^R\}$

45 $\{\langle M \rangle \texttt{\#} w \mid M \text{ accepts } ww^R\}$

46 $\{\langle M \rangle \mid M \text{ accepts } \texttt{HELLO}\}$

47 $\{\langle M \rangle \mid M \text{ rejects } \texttt{HELLO}\}$

48 $\{\langle M \rangle \mid M \text{ accepts at least one palindrome}\}$

49 $\Sigma^* \setminus \{\langle M \rangle \mid M \text{ accepts at least one palindrome}\}$

50 $\{\langle M \rangle \mid M \text{ rejects at least one palindrome}\}$

51 $\{\langle M \rangle \mid M \text{ accepts exactly one string of length } \ell, \text{ for each integer } \ell \geq 0\}$

52 $\{\langle M \rangle \mid \textsc{Accept}(M) \text{ has an infinite fooling set}\}$

53 $\{\langle M \rangle \texttt{\#} \langle M' \rangle \mid \textsc{Accept}(M) \cap \textsc{Accept}(M') \neq \varnothing\}$

54 $\{\langle M \rangle \texttt{\#} \langle M' \rangle \mid \textsc{Accept}(M) \oplus \textsc{Reject}(M') \neq \varnothing\}$ — Here $\oplus$ means exclusive-or.