CPSC 320 2014W2 Final Exam Practice Problems, Video Draft

April 15, 2015

- $\sum_{y=1}^{x} y = \frac{x(x+1)}{2}$, for $x \geq 0$.
- $\sum_{y=1}^{x} y^2 = \frac{x(x+1)(2x+1)}{6}$, for $x \geq 0$.
- $\sum_{y=0}^{x} 2^y = 2^{x+1} - 1$, for $x \geq 0$.

For a recurrence like $T(n) = aT(\frac{n}{b}) + f(n)$, where $a \geq 1$ and $b > 1$, the Master Theorem states three cases:

1. If $f(n) \in O(n^c)$ where $c < \log_b a$ then $T(n) \in \Theta(n^{\log_b a})$.
2. If for some constant $k \geq 0$, $f(n) \in \Theta(n^c (\log n)^k)$ where $c = \log_b a$, then $T(n) \in \Theta(n^c (\log n)^{k+1})$.
3. If $f(n) \in \Omega(n^c)$ where $c > \log_b a$ **and** $af(\frac{n}{b}) \leq kf(n)$ for some constant $k < 1$ and sufficiently large n, then $T(n) \in \Theta(f(n))$.

- $f(n) \in O(g(n))$ (big-O, that is) exactly when there is a positive real constant c and positive integer n_0 such that for all integers $n \geq n_0$, $f(n) \leq c \cdot g(n)$.
- $f(n) \in o(g(n))$ (little-o, that is) exactly when for all positive real constants c, there is a positive integer n_0 such that for all integers $n \geq n_0$, $f(n) \leq c \cdot g(n)$.
- $f(n) \in \Omega(g(n))$ exactly when $g(n) \in O(f(n))$.
- $f(n) \in \omega(g(n))$ exactly when $g(n) \in o(f(n))$.
- $f(n) \in \Theta(g(n))$ exactly when $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

1 Practice Intro

These problems are meant to be generally representative of our final exam problems and—in some cases—may be **very** similar in form or content to the real exam. However, this is **not** a real exam. Therefore, you should not expect that it will fit the predicted exam timeframe or that the questions will be of the appropriate level of specificity or difficulty for an exam. (That is: the real exam may be shorter or longer and more or less vague!)

Furthermore, there are many other practice resources you should attend to—not least our own previous midterms and practice midterms.

All of that said, you would benefit **tremendously** from working hard on this practice exam!

2 Clark Kent's Glasses

Consider these well-known problems (mostly, but not all, NP-complete):

A.	Independent Set	F.	Graph Colouring	K.	Knapsack (p. 267)
B.	Set Packing	G.	Hamiltonian Cycle	L.	Stable Marriage
C.	Vertex Cover	H.	Hamiltonian Path	M.	Sorting
D.	Set Cover	I.	Traveling Salesman	N.	Minimum spanning tree
E.	3-D Matching	J.	Subset Sum		

Each of the problems below is a "disguised" version of one of the problems above. Some problems above may be used multiple times; others may not be used at all. For each problem below, write the letter of the problem above it best matches.

____ The (extremely heatproof) Venus Rover has just landed. Researchers have input a set of sites they'd like the rover to visit. For each pair of sites, mission control has computed the probability of the rover making it between sites without breaking down. They'd like to find a path that visits all the sites while maximizing the chance the rover will remain functional (not break down).

(Note: this one's a bit tricky. In particular, you want to maximize the product of the probabilities along each leg, but what's the log of that quantity?)

____ In your secure cloud computing system, a user rates each other user as "trusted" or "untrusted". Each user is allocated a virtual machine on a single physical machine, but many virtual machines may run on the same physical machine. Given the number of physical machines available, can you allocate everyone to some physical machine without putting two users that distrust each other on the same physical machine?

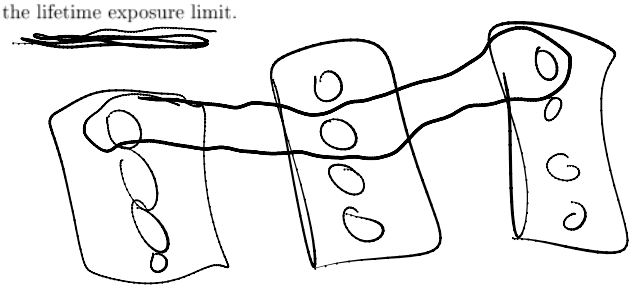
____ A safety program pairs people up across campus into "safety buddies". Some people are in more than one pair, however (depending on their involvement in the program).

The program organizers want to run a training session so that in every group, at least one buddy has been trained. In particular, they'd like to ensure this while keeping the session as small as possible.

____ CS creates a new protocol where students seeking advising rate the advisors, while advisors "bid" on the students they want to advise (effectively also rating them). The system assigns an advisor to each student. The department chooses a load (number of advisees) for each advisor. You want to assign advisors to students to respect ratings and bids as well as you can.

K

The government limits lifetime radioactive exposure for workers. Given the radiation limit and a set of k jobs, each of which has a value and a radiation exposure, find the set of jobs of maximum value that remains below the lifetime exposure limit.



E

You're designing a system to match up groups in the Trimentoring Program, where each group has one mentor, one senior student, and one junior student. You've already pared down the lists so that there is the same number each of mentors, senior students, and junior students. A separate algorithm proposes a list of candidate groups (with potentially many groupings for a particular person). Your algorithm's job is to select a set of groupings among those candidates so that each person is in exactly one trimentoring group.

sets s.t. no two sets overlap

B

You're working for an embedded computing system manufacturer. One of the key concerns they have is configuring how peripherals—extra attached devices—use the input and output ports on the computing devices they build. They've boiled the problem down to the following: A computing device has a labelled list of ports $(1, 2, \ldots, n)$, and they have a list of candidate peripherals, each with a list of ports it must use (e.g., $[1, 3, 8, 9]$). A single port can only be used by one device at a time. Your job is to find the largest set of peripheral devices the computing device can support simultaneously.

3 $O(1)$ Answer Problems

Practice version only: I admit that some of these are not particularly **short**-answer questions. It's my chance to prepare you for other elements of the exam! :)

1. You have a randomized optimization algorithm with a $\frac{1}{n \lg n}$ chance of yielding the optimal solution to a problem. (Each run of this algorithm has the same, indpendent chance of returning the optimal solution.) How many times should you run the algorithm to upper-bound the probability of its failure by $\frac{1}{e}$?

$n \lg n$ times (for $n \geq 2$)

$\left(1 - \frac{1}{n \lg n}\right)^{x} \Rightarrow \left(1 - \frac{1}{n \lg n}\right)^{n \lg n} \leq \frac{1}{e}$ for $n \lg n \geq 1$

$\left(1 - \frac{1}{y}\right)^{y} \leq \frac{1}{e}$ for $y \geq 1$

2. List three reasons you might use randomization in an algorithm.

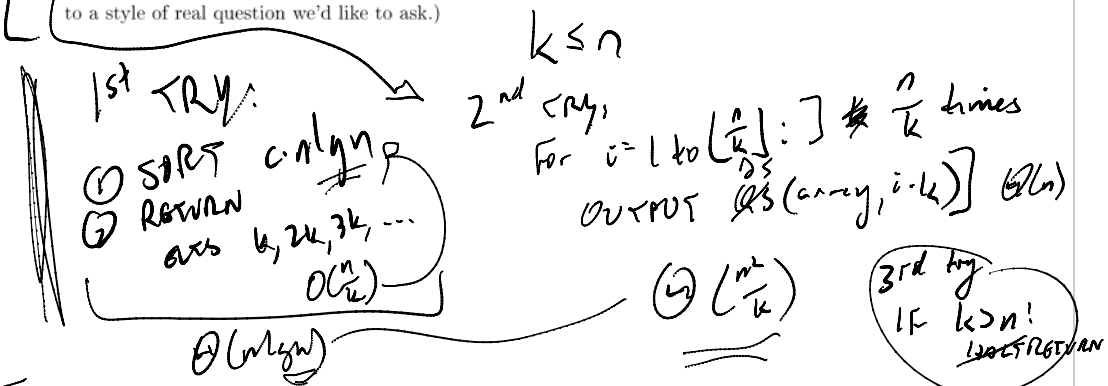
1. Amplify prob. of success of a simple alg.
2. Transition from avg → expected performance (QuickSort)
3. Simplify a complex alg. / practically improve running (QuickSelect)

3. Give five **very different** examples of recurrences to illustrate: the three cases of the Master Theorem and two of the reasons we might not be able to apply the Master Theorem to asymptotically bound a recurrence. Solve each one, showing your work, **including** values for a, b, c, and $f(n)$.

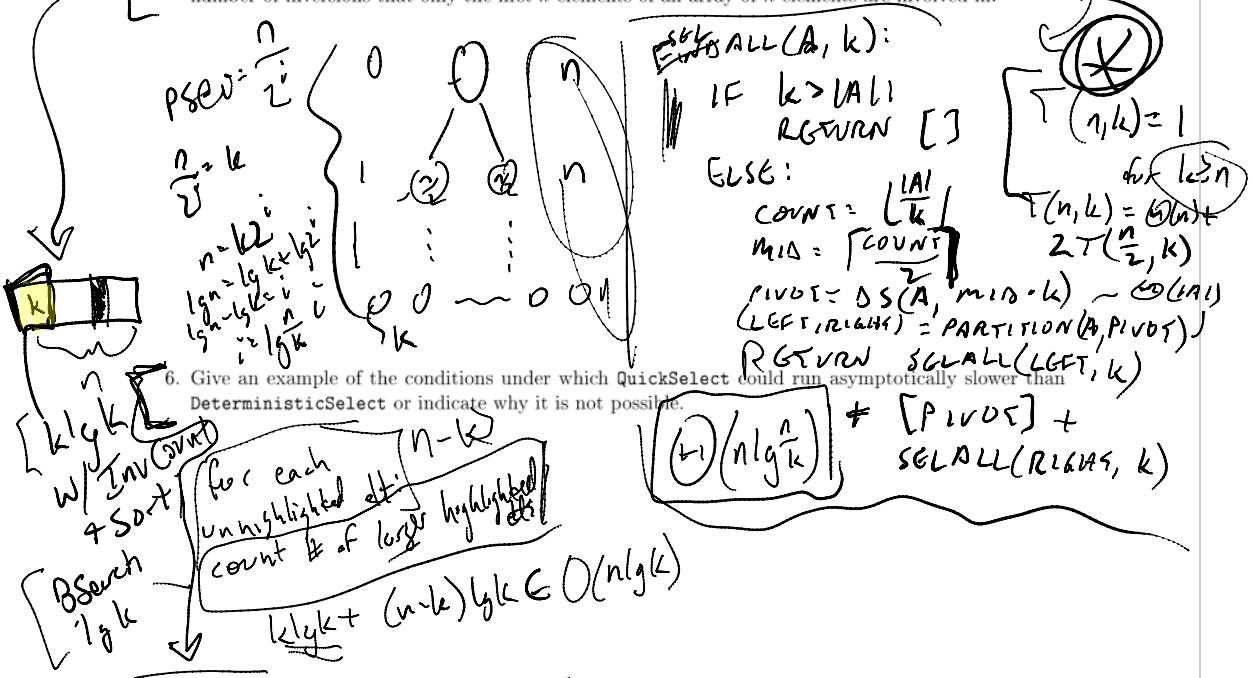
3 leaf root balanced

2 diff. ways that thm is not applicable

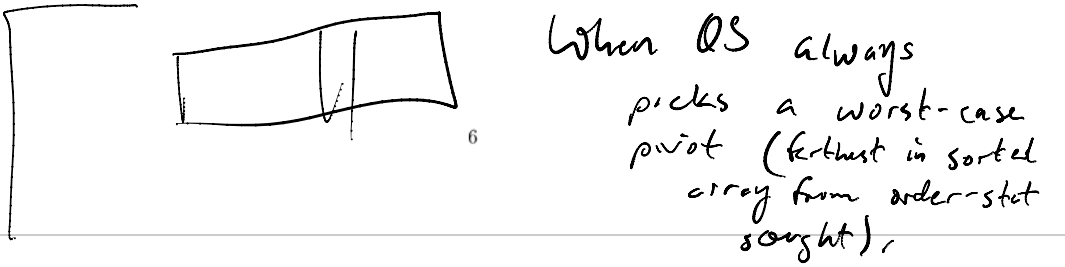
4. Using any existing well-known algorithms or data structures, give an efficient algorithm to find the $k^{\text{th}}, 2k^{\text{th}}, 3k^{\text{th}}, \ldots$ smallest elements of an array of n integers and analyse its runtime in terms of k and n. (Note: this is quite difficult for the short-answer section, but has a somewhat similar flavor to a style of real question we'd like to ask.)



5. Using any existing well-known algorithms or data structures, give an efficient algorithm to count the number of inversions that only the first k elements of an array of n elements are involved in.



6. Give an example of the conditions under which `QuickSelect` could run asymptotically slower than `DeterministicSelect` or indicate why it is not possible.



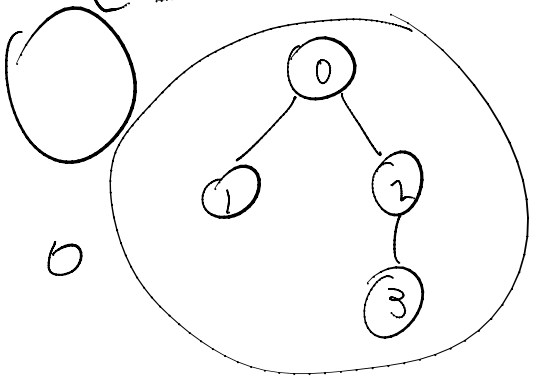
7. A friend claims that there is a much simpler algorithm to find the 4^{th} smallest element in an array of n elements than `DeterministicSelect` in worst-case $O(n)$ time. Give such an algorithm or **briefly** explain why no such algorithm exists.

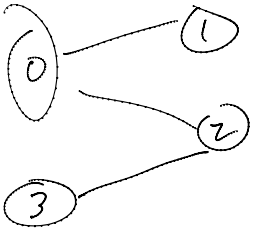
O(n) [O(n) [Find & remove smallest
Repeat 2 times
Find & return smallest.

8. The stable marriage algorithm produces a matching that avoids any instabilities, but instabilities may not be the only factor that makes a matching undesirable. Give **two** other realistic, important reasons the participants in a matching might find it undesirable.

1. Not matched w/ favorite partner (matched w/ highly undesirable partner)
2. Pair's on one side may together demand their optimal result.

9. Consider the statement "A tree is a bipartite graph." Is this true of every tree, some trees, or no trees? **Briefly** justify your answer. (Assume both tree and graph are undirected.)

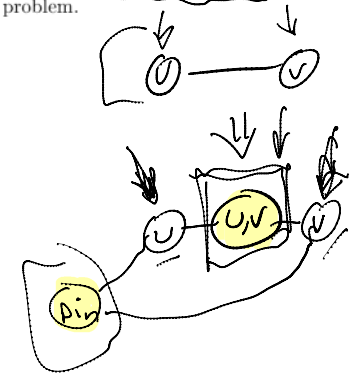




put odd depth nodes on one side & even depth nodes on the other.

10. Consider the following reduction from VC to SP: Given an instance $(G_{VC} = (V_{VC}, E_{VC}), k_{VC})$ of VC, produce an instance $(G_{SP} = (V_{SP}, E_{SP}), S_{SP}, k_{SP})$ of SP (where S_{SP} are the "shaded" nodes and k_{SP} is the maximum number of edges in the Steiner Tree) as follows: for each vertex $v_i \in V_{VC}$, make an unshaded vertex $v_{v_i} \in V_{SP}$; for each edge $(v_i, v_j) \in E_{VC}$, make a shaded vertex $v_{(v_i,v_j)} \in V_{SP}$ and two edges $(v_{v_i}, v_{(v_i,v_j)})$ and $(v_{v_j}, v_{(v_i,v_j)})$ in E_{SP}; make one more shaded vertex $v_{pin} \in V_{SP}$ and connect it with edges to every unshaded vertex; and finally, let $k_{SP} = |E_{VC}| + k_{VC}$.

Briefly explain why any Steiner Tree that solves an SP instance produced by this reduction must connect to one vertex or the other generated from any pair of vertices connected by an edge in the original VC problem.



1 less than $|S_{SP}|$

B/C OTHERWISE THE SHADED NODE REPING THE EDGE BTW VERTS IS NOT CONNECTABLE TO THE PIN.

11. List two distinct reasons why, given two algorithms to solve the same problem, the one with the better worst-case asymptotic performance may not be the best one to use (although it usually is).

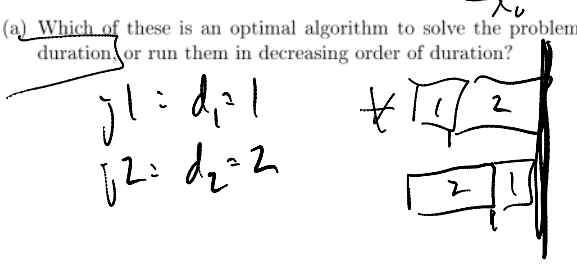
I ASSUME: RUNTIME

(1) PRACTICAL EFFICIENCY (I.E., THE "CONSTANT FACTOR") FOR RELEVANT PROBLEM SIZES.

(2) MEMORY EFFICIENCY!

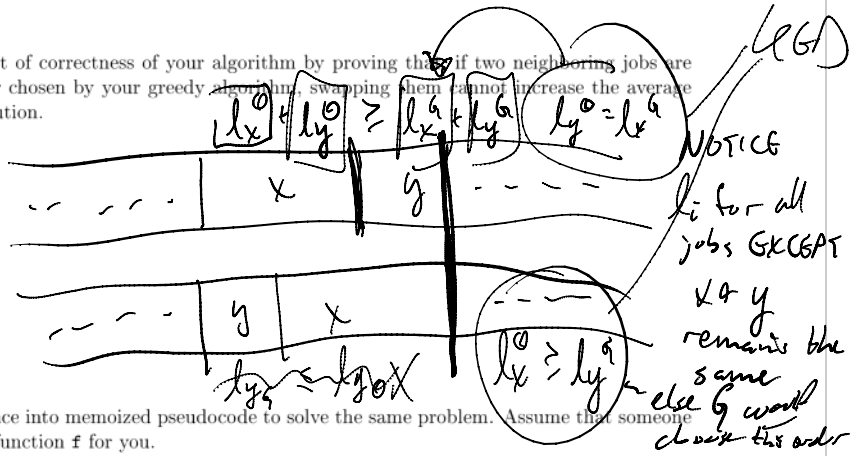
12. You have a single computer to run n jobs. Each job has a duration d_i. You get to order the jobs; they will then be run to completion, one after another, on the single computer. (So, the total time taken will be the same no matter what.) For each job i, the "delay" is the amount of time from the start of running the first job to finishing running job i. (So, the first job's delay is just its duration; the second job's delay is the sum of its duration and the first job's duration; etc.) You want to minimize the average delay across all jobs.

(a) Which of these is an optimal algorithm to solve the problem: run jobs in increasing order of duration, or run them in decreasing order of duration?



(proves not 2nd answer)

(b) Justify the heart of correctness of your algorithm by proving that if two neighboring jobs are not in the order chosen by your greedy algorithm, swapping them cannot increase the average delay of the solution.



13. Convert this recurrence into memoized pseudocode to solve the same problem. Assume that someone else has written the function `f` for you.

$$D(i,n) = \begin{cases} 0 & \text{when } i < 1 \text{ or } n < i \\ D(i-1,n-1) * D(i-2,n-i) + f(i,n) & \text{otherwise} \end{cases}$$

naïve

```
D(i,n):
  if i<1 or n<i:
    RETURN 0
  else:
    RETURN D(i-1,n-1)*D(i-2,n-i)+f(i,n)
```

mem.

```
D(i,n):
  ALLOC TABLE[i][n]
    NAMES SOLN INITD TO null
  Dmem(i,n,SOLN)

Dmem(i,n,SOLN):
  if i<1 or n<i:
    RETURN 0
  else if SOLN[i][n] is null:
    SOLN[i][n] = Dmem(i-1,n-1,SOLN)*Dmem(i-2,n-i,SOLN)+f(i,n)
  RETURN soln[i][n]
```

comp sol'n & store in table

14. For the recurrence from the previous problem, assuming that `f(i, n)` runs in $O(i \lg n)$ time, give (but do not solve) a recurrence $T(m,n)$ for the time it takes to compute `D(m, n)` without memoization (including all appropriate cases!).

$$T(m,n) = \begin{cases} 1 & \text{when } i<1 \text{ or } n<i \\ T(n-1,n-1)+T(n-2,n-n)+O(n \lg n) & \text{otherwise} \end{cases}$$

15. True or False:

- Even if P = NP, some algorithmic problems may not be solvable in polynomial time.

15. True or False:

- Even if P = NP, some algorithmic problems may not be solvable in polynomial time.

  TRUE **FALSE**

- When we say "a problem of size n can be solved in polynomial time", we mean "all algorithms for solving the problem take $O(n^c)$ time in the worst case, for some constant $c > 0$."

  TRUE **FALSE**

- If P ≠ NP, then each instance of an NP-complete problem takes more than polynomial time to solve.

  TRUE **FALSE**

- Memoizing an algorithm only helps if the algorithm solves the same subproblem multiple times.

  TRUE **FALSE**

- The performance recurrences for binary search and mergesort both illustrate examples of the "balanced" case of the Master Theorem.

  TRUE **FALSE**

- An adversary can provide input that forces worst-case performance from a version of QuickSort that chooses the median of the first element, the middle element, and the last element as its pivot.

  TRUE **FALSE**

- There is an efficient algorithm known to find the minimum spanning tree of an undirected graph with integer (and possibly negative) edge weights.

  TRUE **FALSE**

- There is an efficient algorithm known to find the longest path between two nodes in a directed graph with postive edge weights.

  TRUE **FALSE**

4 Some Probes Are More Equal than Others

A set of n points (for integer $n > 0$) numbered $1, 2, 3, \ldots, n$ are arranged in an array.

You are given n and an operation `probe` to access the points. Given a point number `p`, `probe(p)` returns `p`'s associated value.

However, probing a point `p` has a cost `cost(p)`. (Checking the cost of a point costs nothing, and probing a point more than once also costs nothing.)

The points are sorted in increasing order by their associated values; so, if i and j are point numbers with $i < j$, then `probe(i)` < `probe(j)`.

You would like to **find a target value** t in such a way that you minimize the worst-case total cost of your probes.

1. Here is a modest-sized instance of the problem:

index:	0	1	2	3	4	5	6
value:	2	7	8	13	15	25	90
cost:	20	20	24	10	9	40	10

   The minimal worst-case cost of 60 is achievable using a standard binary search approach. **Briefly** explain what target generates the worst-case cost and why.

2. Give at least one trivial and one small instance and their minimal worst-case total probe costs.

3. Give an instance—and its minimal worst-case total probe cost—with 5 points that illustrates that binary search is not an optimal approach in general.

4. Complete this recurrence for the minimum worst-case total probe cost for the subarray `A[i..j]`, i.e., the subarray that includes indexes $i, i+1, \ldots, j-1, j$:

$$C(i,j) = \begin{cases} min_{i \le k \le j}(&) \quad \text{when} \\ & \quad \text{otherwise} \end{cases}$$

5. Write pseudocode for a memoized version of this algorithm. Be sure to indicate clearly the size of table you will need and any initialization the table requires.

6. Write pseudocode for a dynamic programming version of this algorithm. Be sure to indicate clearly the size of table you will need and the order in which you will solve subproblems.

7. Asymptotically analyse the runtime and memory usage of the memoized solution in terms of n.

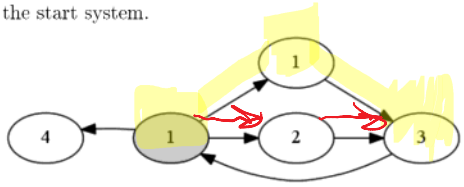
8. Assuming you **only** want to know the minimum worst-case cost of solving the problem (not the actual sequence of probes to make), asymptotically analyse the memory usage of an efficient dynamic programming solution.

5 Empirical Algorithms

You're designing a space game in which a single player explores a set of star systems in the universe. Some pairs of star systems are connected by wormholes; others are not. A wormhole can only be traveled in a **single** direction. The player can complete a valuable quest at each star system. Because the player is a fugitive closely followed by the evil Empirical Forces, they can never revisit a star system once they leave it. You've written an algorithm to generate an attractive "universe" (collection of star systems and wormhole connections), but you're stuck on how to ensure that the universe allows for a large enough score for the player.

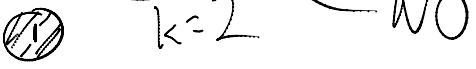
In particular, you want to solve the Space Quest problem SQ: Given a list of the star systems, a list of the wormhole connections, a starting star system for the player, an integer value for each star system's quest, and a "minimum quest result" k, is there a path through the universe starting at the start system with total value of quests at each system of at least k?

1. Here is an instance of SQ. Circles are star systems; each system's integer quest value is inside its circle. Arrows are wormholes (which can be traveled in the direction of the arrow). The shaded system is the start system.

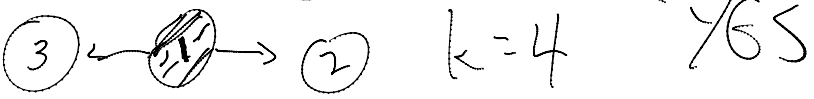


Its solution is YES for $k = 5$. Circle or shade the wormhole arrows followed to yield a path that respects the problem's constraints.

2. Give a trivial instance of SQ and its solution.



3. Give a small instance of SQ and its solution.



4. Prove that SQ $\in$ NP.

Let cert be the path that will accum k or more quest value. — poly length as list of nodes.

$p_1, p_2, p_3, \ldots, p_t$

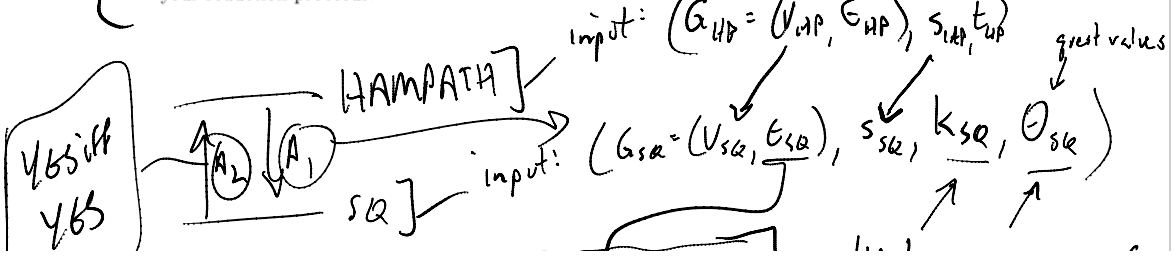
① Ensure $(p_i, p_{i+1}) \in E$ for all $1 \le i < t$. — poly time

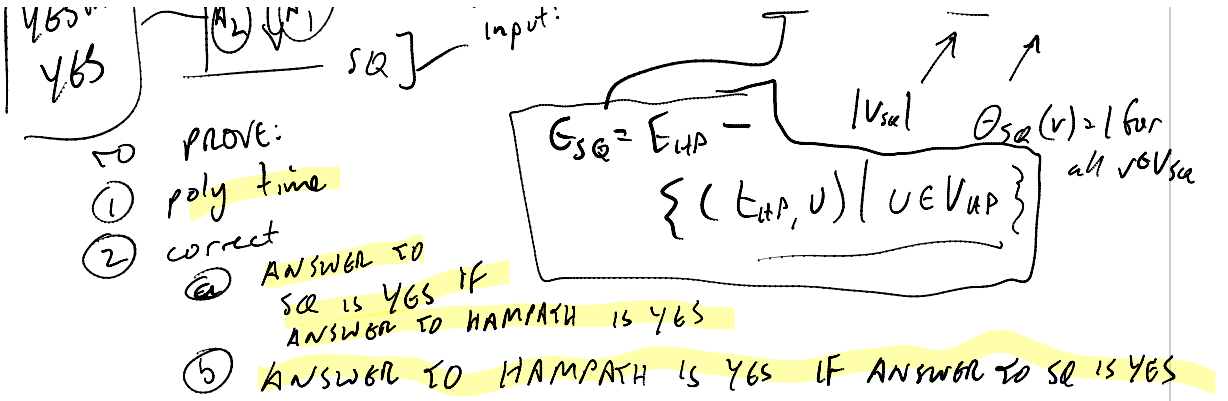
② ensure p_1 is start system — $O(1)$

③ Ensure no dups in $p_1, \ldots, p_t$ — poly time

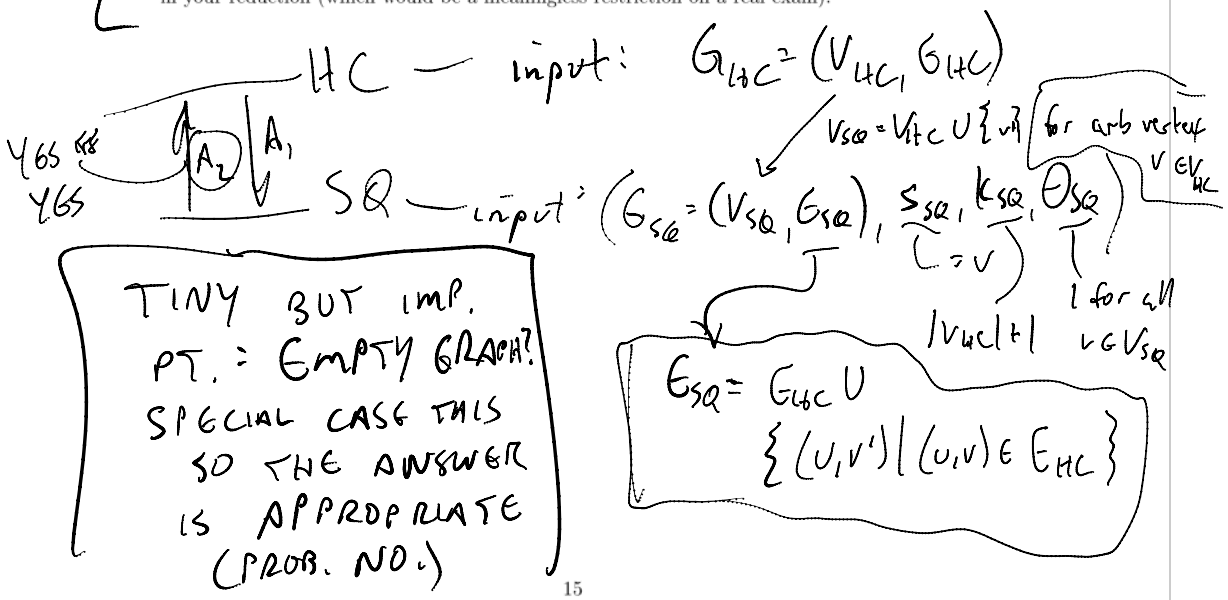
④ Ensure $\sum_{i=1}^{t} \text{value}(p_i) \ge k$ — poly time

5. The Hamiltonian Path problem (HAMPATH, on directed graphs) is known to be NP-complete. Prove using a reduction involving HAMPATH that SQ $\in$ NP-hard. Be sure to clearly indicate which direction your reduction proceeds! — Give





6. The Hamiltonian Cycle problem (HAMCYCLE, on directed graphs) is known to be NP-complete. Prove using a reduction involving HAMCYCLE that SQ $\in$ NP-hard. Be sure to clearly indicate which direction your reduction proceeds! Please do not use HAMPATH (or any other problem) as an intermediate step in your reduction (which would be a meaningless restriction on a real exam).



7. **Prove** that one of your reductions is correct and takes polynomial-time.

ALREADY REQ'D BY "PROVE" ABOVE.

WATCH 5.5 VIDEO.

8. A friend proposes a reduction from SQ to HAMPATH that includes this step: "for a vertex v in SQ labeled with quest value q, generate q vertices $v_1, \ldots, v_q$ in HAMPATH...".

   Explain why such a reduction does not qualify as the kind of "polynomial-time reduction" we've been using in our NP-completeness proofs.

?

creates # of verts proportional to q.

say each vertex is involved in an edge s.t. you need ≥ 1 bit per edge.

How MANY BITS DOES IT TAKE TO REP. q?

$O(\lg q)$

SO: UNDERLYING INSTANCE'S LENGTH MAY BE MORE THAN POLYNOMIAL.

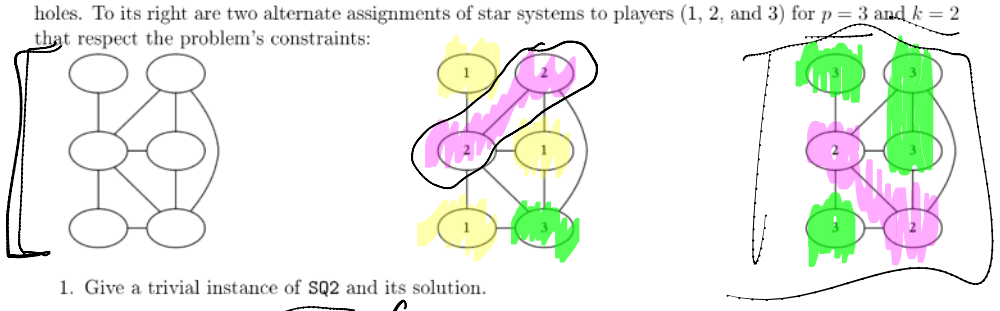
6 Empirical Algorithms Strike Back

You're designing a sequel game in which a group of players each controls a Spaaaaace Empire: a set of star systems in the universe. Some pairs of star systems are connected by wormholes; others are not. (Wormholes can be traveled in **either** direction, but no travel between star systems is possible without a wormhole.) You've written an algorithm to generate an attractive "universe" (collection of star systems and wormhole connections), but you're stuck on how to divide the universe up among the players when the game starts.

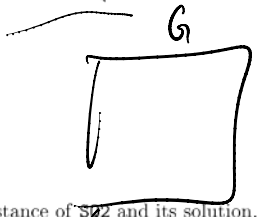
In particular, you want to solve the Space Quest 2 (SQ2) problem: Given a list of the star systems, a list of the wormhole connections, a count of players p, and an "empire size" k, can you assign each star system to exactly one player while ensuring that no player controls any connected set of star systems larger than k?

Note: Players may start with no star systems at all. The universe must be connected.

On the left below is a modest-sized instance of SQ2, where circles are star systems and lines are wormholes. To its right are two alternate assignments of star systems to players (1, 2, and 3) for $p = 3$ and $k = 2$ that respect the problem's constraints:

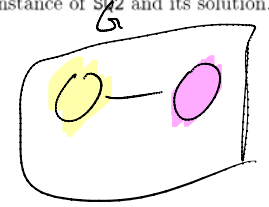


1. Give a trivial instance of SQ2 and its solution.



p=0 k=0
ANSWER: YES

2. Give a small instance of SQ2 and its solution.



p=2 k=1
ANSWER: YES

3. Prove that SQ2 ∈ NP.

certificate: f that maps each vertex to the player that controls it.
list for each vertex of player # who controls it

(1) ENSURE THE LIST HAS |V| ENTRIES ← poly
(2) ENSURE EACH ENTRY IS IN 1...p. ← poly
(3) FOR EACH VERTEX v: ← |V| poly
BFS FROM v WITHOUT ALLOWING VISITS TO NODES OWNED BY OTHER PLAYERS (BESIDES v's owner), COUNTING # OF UNIQUE NODES VISITED
ENSURE THAT IF ≤ k.

4. Give a reduction from graph colouring (GC) to this problem. Assume the GC graph is connected.

Note: For this and **all subsequent parts**, "reduction" means "polynomial-time reduction", but you need not show your reduction takes polynomial time.

"YES iff YES"
GC
GC input: $(G_{GC} = (V_{GC}, E_{GC}), k_{GC})$
A_1
A_2
SQ2
SQ2 input: $(G_{SQ2} = (V_{SQ2}, E_{SQ2}), p_{SQ2}, k_{SQ2})$

SQ_2 SQ_2 input: $(G_{SQ2} = (V_{SQ_2}, E_{SQ_2}), P_{SQ_2}, k_{SQ_2})$

5. You decide to modify SQ2 to restrict the number of players so $p \leq 3$. Is the problem still NP-complete? Briefly justify your answer.

GC w/ $k=3$ IS NP-COMPLETE.
SO, THE REDUCTION ABOVE SHOWS
SQ_2 w/ $p \leq 3$ IS NP-HARD.
+ IT'S OBVIOUSLY STILL IN NP.

6. Once more you go back and modify the original problem SQ2 to require that the universe—viewed as a graph—is bipartite.

   In that case, for what values of p and k can you guarantee that the solution to the problem is YES? **Briefly** justify your answer.

For $p \geq 2$, THE GRAPH IS "2colorable"!
For $p = 1$, ONLY IF $k \geq$ |largest conn. comp| (connected G: |V|)
For $p = 0$, ONLY FOR empty graph.