Department of Electrical Engineering, National Taiwan University
Algorithms, ~~Fall~~ Spring 2019

April 26, 2019
Iris Hui-Ru Jiang

EE4033; 901/39000: Midterm

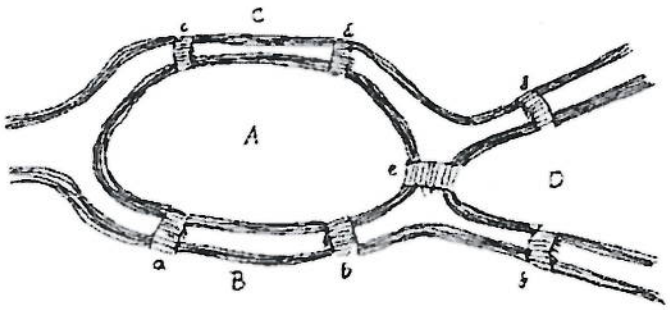
1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
/6	/6	/8	/6	/6	/24	/14	/10	/14	/6

Name: _____ **ID:** _____

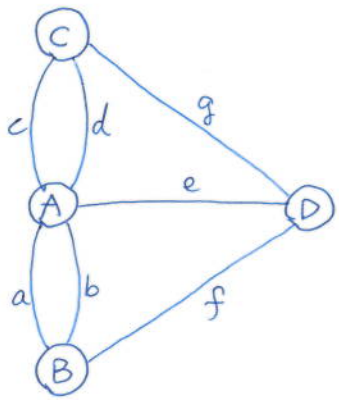
Instructions:

1. This is a 180-minute closed-book test graded out of 100, 10 problems.
2. You are allowed to refer one A4-size cheat sheet with only hand-written notes.
3. Please **hand in your cheat sheet and problem sheets with your solutions** for avoiding penalty.
4. Pace yourself! Good luck!

1. **(6 pts)** Salute to Leonhard Euler!
Background: In the early 18th century, the citizens of Königsberg spent their days walking on an intricate arrangement of bridges across the waters of the Pregolya River, which surrounded two central land masses (*A* and *D* in the following map) connected by a bridge. The first landmass (an island) was connected by four bridges, with two bridges connecting the lower bank (*B*), and two bridges connecting the upper bank (*C*). The other landmass (*D*), which split the Pregel into two branches, was connected to the lower and upper bank by two bridges, with one other bridge connecting both landmasses together. In total, there were seven bridges (*a*, *b*, *c*, *d*, *e*, *f*, *g* in the map).
Question: According to folklore, a question arose: *could a citizen take a walk through the town in such a way that each bridge would be crossed exactly once? i.e., is it possible to start at some place in the town, cross every bridge exactly once, and return to the starting place?* Model this problem as a graph problem. Define vertices and edges in your graph and draw the corresponding graph model for the following Königsberg map. (You do not need to answer the above question.)



Euler's Drawing of Konigsberg Bridges.

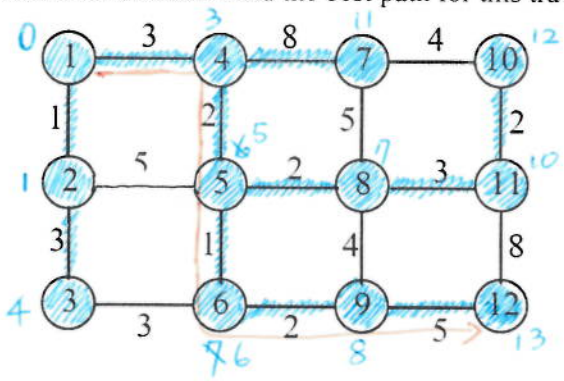


$G = (V, E)$
vertex $\in V$: landmass or bank.
edge $\in E$: bridge.

Find a cycle (path) starting with a vertex, through every edge exactly once, and returning to the starting vertex.

Graph (+2)
Define V, E (+2)
Model problem (+2)

2. **(6 pts)** Think of the network shown below as a highway map, and the number recorded next to each arc as the maximum elevation encountered in traversing the arc. A traveler plans to drive from node 1 to node 12 on this highway. This traveler dislikes high altitudes and so would like to find a path connecting node 1 to node 12 that minimizes the maximum altitude. Find the best path for this traveler.



Run Dijkstra's algorithm and set node 1 as source s, node 12 as target, destination.
and elevation is the edge cost/weight.

The best path is the shortest path from node 1 to node 12:

<1, 4, 5, 6, 9, 12>

The maximum altitude is 13.

Correct path (+6)

Partial:
Dijkstra (+3)

3. **(8 pts)** Order the following functions from the slowest (with the lowest complexity) to fastest growth.

$n^{3/2}$, $\sqrt{n}(\lg n)^3$, $(\sqrt{n})^{\lg n}$, $(\lg n)!$

Take $\lg$ ($\log_2$) on each function

① $\lg(n^{3/2}) = \frac{3}{2}\lg n$

② $\lg(\sqrt{n}(\lg n)^3) = \lg\sqrt{n} + 3\lg(\lg n) = \frac{1}{2}\lg n + 3\lg(\lg n)$

③ $\lg((\sqrt{n})^{\lg n}) = \lg n \cdot \lg\sqrt{n} = \frac{1}{2}(\lg n)^2$

④ $\lg(\lg n)! = \lg((\lg n)(\lg n - 1)\cdots) \leq \lg \underbrace{\lg^n \cdot \lg^n \cdots}_{\lg n} = \lg(\lg n)^{\lg n}$

$= \lg n \cdot \lg(\lg n)$

∴ ② < ① < ④ < ③

i.e., $\sqrt{n}(\lg n)^3 < n^{3/2} < (\lg n)! < (\sqrt{n})^{\lg n}$

Correct (+8)
Each mistake (-2)

4. **(6 pts)** In lecture, we discussed the simple bound of BUILD-MAX-HEAP is O(*n*log*n*). Here, please give a **tight** analysis for the time complexity of BUILD-MAX-HEAP listed below.

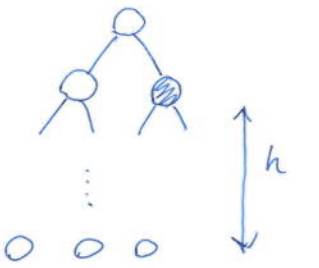
```
BUILD-MAX-HEAP(A)
1. A.heap-size = A.length
2. for i = A.length/2 downto 1
3.     MAX-HEAPIFY(A, i)
```

Here, MAX-HEAPIFY(A, i) will float down the value at $A[i]$ so that the subtree rooted at $A[i]$ becomes a heap, given that the two subtrees of $A[i]$ are heaps.

Each MAX-HEPIPY takes $O(h)$ time (h: height)

There are at most $\lceil n/2^{h+1} \rceil$ nodes of height h in array A. (+2)

(n means the total number of nodes)



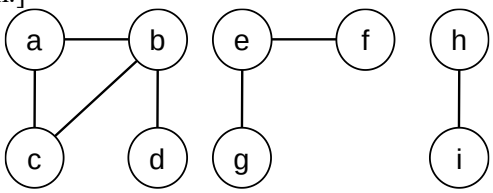
$$T(n) = \sum_{h=0}^{\lfloor \lg n \rfloor} (\#\text{nodes of height } h) \cdot O(h) \quad (+3)$$

$$\leq \sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil \cdot O(h) = O\left(n \cdot \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) = O(n) \quad (+1)$$

PS. $\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{\frac{1}{2}}{(1-\frac{1}{2})^2} = 2$

5. **(6 pts)** Find connected components in the following graph by utilizing **union-find** (disjoint set) data structure. Three operations are supported:
   - MakeUnionFind(*S*): initialize a set for each element in *S*
   - Find(*u*): return the representative of the set containing *u*
   - Union(*A*, *B*): merge sets *A* and *B*

   Please fill the following table to show the step-by-step collection of disjoint sets every time we process an edge. [Hint: Kruskal's Algorithm.]



Edge processed	Collection of disjoint sets
- (Initial sets)	{a}, {b}, {c}, {d}, {e}, {f}, {g}, {h}, {i}
(b, d)	{a}, {b, d}, {c}, {e}, {f}, {g}, {h}, {i}
(e, g)	{a}, {b, d}, {c}, {e, g}, {f}, {h}, {i}
(a, c)	{a, c}, {b, d}, {e, g}, {f}, {h}, {i}
(h, i)	{a, c}, {b, d}, {e, g}, {f}, {h, i}
(a, b)	{a, b, c, d}, {e, g}, {f}, {h, i}
(e, f)	{a, b, c, d}, {e, f, g}, {h, i}
(b, c)	{a, b, c, d}, {e, f, g}, {h, i}

Correct (+6)
Each wrong (Initial, notation, mistake) (-2)

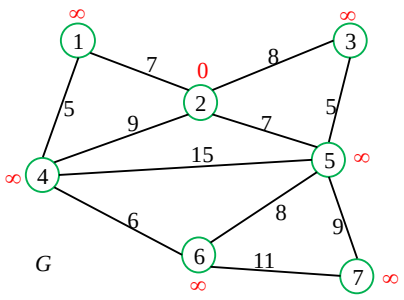
6. **(24 pts)** Given a weighted connected undirected graph $G=(V, E)$, where $c(u, v) \geq 0$ represents the length between nodes u and v. Dijkstra's algorithm listed as follows can find the shortest path from node s to any other reachable node.

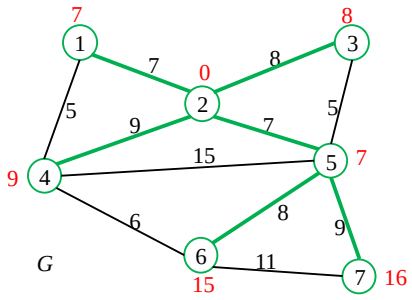
Each (+4)
(a)(e) Yes/No (+2) Why (+2)
(c)(f) Before (+1) After (+3)

```
Dijkstra(G, s, c)
1. initialize S = {s}, d(s) = 0
2. while S ≠ V do
3.     select a node v∉S with at least one edge from S for which
           d'(v) = min_{(u, v) ∈ E: u∈S, v∉S} {d(u) + c(u, v)}
4.     add v to S and define d(v) = d'(v)
```

(a) Is the constraint "$c(u, v) \geq 0$" necessary for Dijkstra's algorithm? Why?
(b) If each edge has the same weight, please briefly describe how to find a linear-time algorithm instead.
(c) Execute Dijkstra(G, s, c) on the given graph G, s = 2. Suppose $d(v)$ for each node v is maintained by a min heap. Show the value of $d(v)$ for each node right after line 1 is executed and after the algorithm terminates. Highlight the shortest path spanning tree edges. ($d(v)$ is better than $d'(v)$)
(d) Transform Dijkstra's algorithm to Prim's algorithm so that Prim(G, s, c) can find a minimum spanning tree containing node s. (Indicate the line number and revised content.)
(e) Is the constraint "$c(u, v) \geq 0$" necessary for Prim's algorithm? Why?
(f) Execute Prim(G, s, c) on the same graph G, s = 2. Suppose $d(v)$ for each node v is maintained by a min heap. Show the value of $d(v)$ for each node right after line 1 is executed and after the algorithm terminates. Highlight the minimum spanning tree edges.

(a) Yes. Line 3 will be wrong for negative edges (cycles), since the shortest path length is accumulated.
(b) BFS.
(c) In the sample solution, ∞ could be replaced by a large enough number.

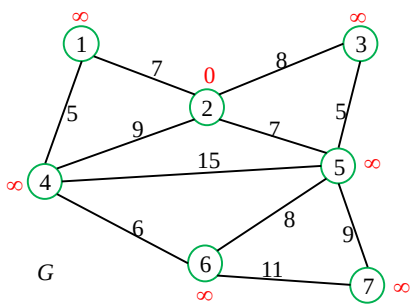


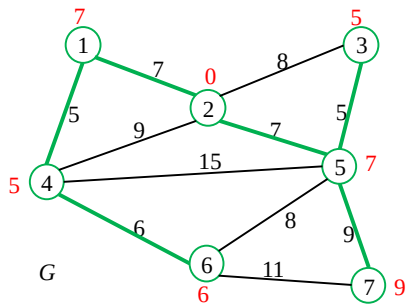


(d) Line 3 is modified.

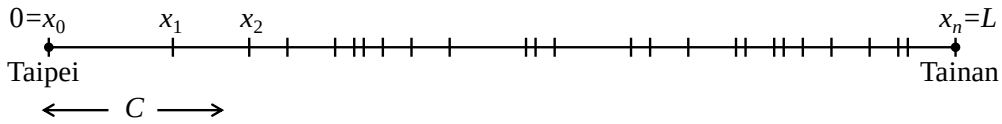
```
Prim(G, s, c)
1. initialize S = {s}, d(s) = 0
2. while S ≠ V do
3.     select a node v∉S with at least one edge from S for which
           d'(v) = min_{(u, v) ∈ E: u∈S, v∉S} c(u, v)
4.     add v to S and define d(v) = d'(v)
```

(e) No. line 3 depends only on the currently selected edge.
(f)





7. **(14 pts)** Sophie plans to drive from Taipei to Tainan along the Formosa highway for her best friend's wedding banquet. The distance from Taipei to Tainan along the way is L kilometers. Her car with full fuel can travel C kilometers. Her google map gives the locations of refueling stations on her way, b_0, b_1, b_2, …, b_n. b_0 is the starting point ($x_0 = 0$), b_n is the end point ($x_n = L$), and $x_{i+1} - x_i \le C$ for $0 \le i < n$.



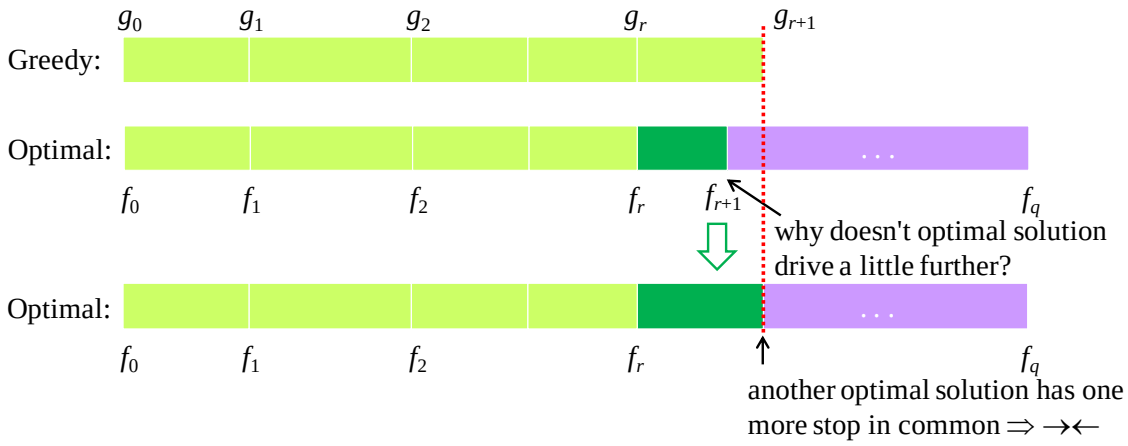
Her car's gas tank is full at x_0. She wishes to make as few refueling stops as possible for this route. Give an optimal algorithm to determine the refueling stations she should stop. Prove your algorithm is optimal. Partial credits for an algorithm with a higher time complexity.

(a) Given refueling stations sorted in ascending order.

```
Driver(X, L, C)
// X = {x0 < x1 < x2 < ... < xn }: refueling stations
1. S = {0}              // refueling stations selected
2. x = 0                // current location
3. while (x ≤ xn) do
4.     p = the largest integer such that xp ≤ x + C
5.     if (xp = x) then
6.         return "no solution"
7.     x = xp
8.     S = S + {p}
9. return S
```

Algorithm (+7)
Prove (+7)

PS. Line 1: 0 can be omitted.
Running time: O(n) (line 4 scans all x_i only once to select each possible stop p.)
(b) Proof by contradiction.
Assume greedy is not optimal, and let's see what happens.
Let $0 = g_0 < g_1 < \ldots < g_p \le L$ denote the set of stop locations chosen by greedy.
Let $0 = f_0 < f_1 < \ldots < f_q \le L$ denote the set of stop locations in an optimal solution with $f_0 = g_0$, $f_1 = g_1$, …, $f_r = g_r$ for a largest possible value of r.
Note: $g_{r+1} > f_{r+1}$ by greedy choice of algorithm. (see line 4)



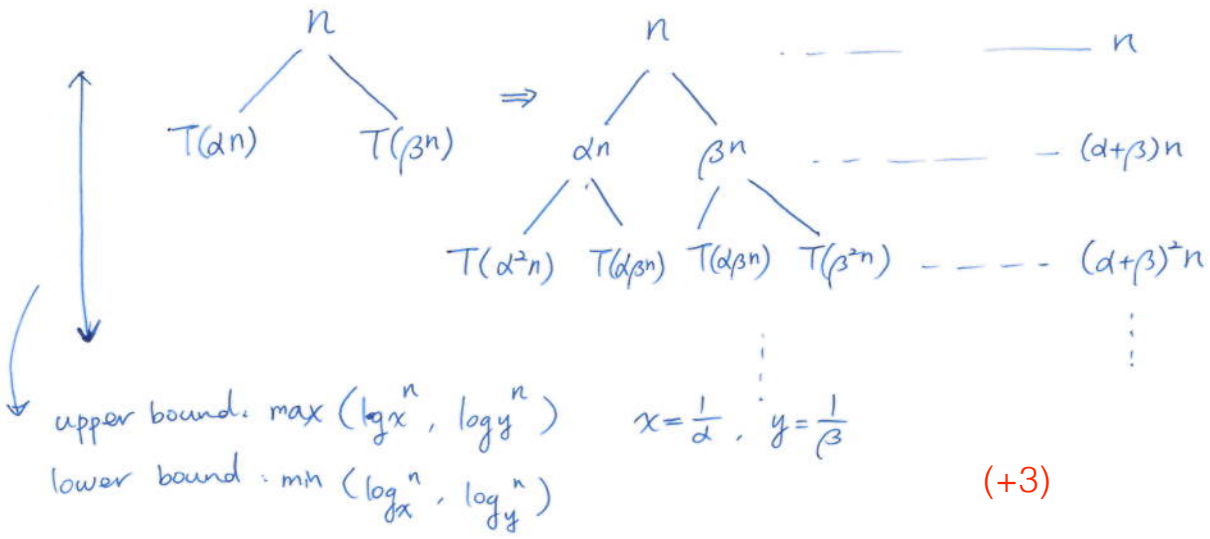
8. **(5+5 pts)** For each of the following recurrence relations, give the asymptotic growth rate of the solution using the θ notation. Assume in each case that $T(n)$ is $\theta(1)$ for $n \le 2$.
   (a) $T(n) = 3\,T(n/2) + n / (\log_2 n)$. Show your steps.
   (b) $T(n) = T(\alpha n) + T(\beta n) + n$, where $\alpha + \beta < 1$. Show your steps.

Each (+5)

(a) $T(n) = 3T(n/2) + n/\log_2 n = \theta(n^{\log_2 3})$ i.e., $\theta(n^{\lg 3})$

$f(n) = n/\log_2 n = O(n^{\log_b a - \epsilon}) = O(n^{\log_2 3 - \epsilon})$ case 1 of Master Theorem applies.

(b) Construct the recursion tree $T(n) = T(\alpha n) + T(\beta n) + n$, $\alpha+\beta<1$.



upper bound: $\max(\log_x n, \log_y n)$ $x=\frac{1}{\alpha}$, $y=\frac{1}{\beta}$

lower bound: $\min(\log_x n, \log_y n)$ (+3)

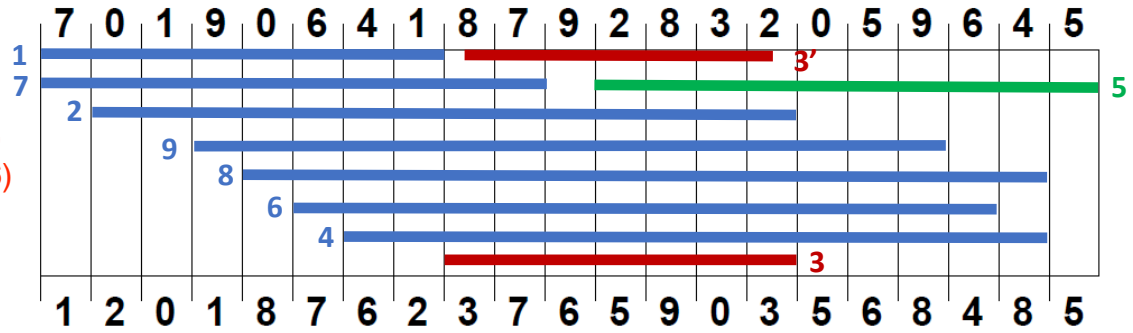
$\sum_{k=0}^{\max(\log_x n, \log_y n)} (n \cdot (\alpha+\beta)^k) = O(n)$ (+1)

$\sum_{k=0}^{\min(\log_x n, \log_y n)} (n \cdot (\alpha+\beta)^k) = \Omega(n)$

$\therefore T(n) = \Theta(n)$ (+1)

9. **(10+4 pts)** A channel is a horizontal routing area with fixed pins on the top and bottom. There are no pins to the right or left, but certain nets may be designated as route-to-the-edge. Thus, we say that there may be "floating pins" to the right or left. Each pin is designated by a number. All pins with the same number must be routed together. Pins with different numbers must be electrically isolated from one another.
The input to a channel routing problem is two sets of numbers, one that gives the pin numbers at the top of the channel, and the other that gives the pin numbers at the bottom of the channel. A pin-number of zero designates an empty pin that is never connected to anything. The following shows an example of a channel, where the only important thing is the numbers.

Example (+4)
Algorithm (+6)
Prove (+4)



Channels are routed in three layers, one layer for the horizontal wires, two layers for the vertical. Except where contact-cuts are created, all layers are insulated from one another. The collection of all pins with the same number is called a net. Each net is given a single horizontal segment which runs from the first occurrence of a pin to the last occurrence. Vertical segments are added to connect the horizontal segment to the pin. Given n nets, please devise an O($n$log$n$)-time algorithm to complete a channel routing instance using the least number of tracks. Prove your algorithm reaches the lower bound. One track contains non-overlapping horizontal segments. Complete the example shown above based on your algorithm.

Sol: Interval coloring (leftedge algorithm)
(a) Each signal is represented by an interval (running from the first occurrence of a pin to the last occurrence)
(b) The number of tracks needed is at least the depth d of the set of intervals (lower bound).(+2)
(c) Different signals cannot occupy the same columns at the same track **3'**
Or, different signals cannot occupy adjacent columns at the same track (at least one empty column in between them **3**)

```
Interval-Partitioning(R)
1. {I_1, …, I_n} = sort intervals in ascending order of their start columns   (+2)
2. for j from 1 to n do
3.     exclude the labels of all assigned intervals that are not compatible with I_j
4.     if (there is a nonexcluded label from {1, 2, …, d}) then
5.         assign a nonexcluded label to I_j                                  (+4)
6.     else leave I_j unlabeled
```

Pf:

(+2)

1. No interval ends up unlabeled.
   - Suppose interval I_j overlaps t intervals earlier in the sorted list.
   - These $t + 1$ intervals pass over a common point, namely $s(I_j)$.
   - Hence, $t + 1 \le d$. Thus, $t \le d - 1$; at least one of the d labels is not excluded, and so there is a label that can be assigned to I_j.

   i.e., line 6 never occurs!
2. No two overlapping intervals are assigned with the same label.
   - Consider any two intervals I_i and I_j that overlap, $i < j$.
   - When I_j is considered (in line 2), I_i is in the set of intervals whose labels are excluded (in line 3).
   - Hence, the algorithm will not assign the label used for I_i to I_j.
3. Since the algorithm uses d labels, we can conclude that the greedy algorithm always uses the minimum possible number of labels, i.e., it is optimal!

10. **(6 pts)** Consider a town with n men and n women seeking to get married to one another. Each man (woman) has a preference list that ranks all the women (men). The set of all $2n$ people is divided into two categories: *good* people and *bad* people. Suppose there are k good men and k good women, and thus there are n-k bad men and n-k bad women, $1 \le k \le n$-1. Everyone would rather marry any good person than any bad person, i.e., in each preference list, its first k entries are good people of the opposite gender in some order, and its next n-k are bad people of the opposite gender in some order. Show that in every stable matching, every good man is married to a good woman.

Sol:

Solved ex.1 in chapter 1. Proof by contradiction.

Idea: Assume the claim is false and try to work toward obtaining a contradiction.

Suppose there exists some stable matching M in which a good man m_g was married to a bad woman w_b. (+2)

There are k good men and k good women.
One good men (m_g) is married to a bad woman (w_b), that leaves only k-1 other good men, even if they all were married to good women, that is still some good woman (w_g) married to a bad man (m_b).

We can identify an instability (m_g, w_g) in M, which contradicts our assumption that M is stable. (+4)