Cpt S 223
Program 4 (Key) Total points = 100

Problem #1 Group Cities By Country

Input: <city, country> ⋮ list (unordered) → (code) → Output: Country 1 (all cities in that country), Country 2 (all cities here) ⋮

Main Design Idea/Options:

→ **Option I:** (Best Solution)

Sort Input <city, country> list by country field
- → (because this is string sorting they have to write their own comparison function)
- → Any $O(n \lg n)$ sort will be good — merge sort — quicksort

→ **Option II:** (Equally good as Option I)

Hash country ~~to~~ → key. ~~But the And then sort the Key values~~. Then Use Chaining $O(n + M)$ answer [M = size of hashtable]

~~This is a bad solution because key value can range over arbitrary range~~ ~~and it is as bad as the Option I, atleast sort based algo~~

→ **Option III:**

Use Union-find to group country names.

<city, ~~country~~> ⇒ Index/Hash city name to ~~country~~ a number ~~city~~ from 1 to n → Build Union find over n cities
↓
For every pair of cities see if their country is the same
↓
If so, then Union(city₁, city₂)

Complexity $= O(n \times \text{hashing}) + O(n^2 \lg^* n)$ → $O(n^2)$ loop

Bad solution because more complicated than necessary.

Problem #1 Grading rules

Total points for prob #1 } = 50

25 Design Choice
- Sorting (option I) or (Hashing Option II) → 25
- Any other design ~~that~~ is ~~wrong~~ → ≤15

25 Implementation
- Code Readability → 5
- Does the code work (testing)? → 20

If Report is missing or is totally uninformative, then ~~to~~ reduce 10 points for that

Problem #2 Group cities by Air Routes (50 points)

Input: <city A, city B> → Code → Output: Eq. Class #1 A, B, C, D

means A & B are connected by flight

Eq. Class #2 E, X, Y, ...

Goal: If 2 cities have an air route/path between them, then they belong to the same Eq. Class.

Design Choices:

→ Option I: Use Union+find thrashing

Eg: <A, B> <C, D> <D, A> <E, F> → For * line in Input <X, Y> do: hash(X), hash(Y) → check ~~hash(Y)~~ Do Union(hash(X), hash(Y)) → Report all Union-find groups/classes

main problem: Union-find will have hash table sized array - O(m) → could become very big

→ Option II:

Just Use Union-find & then for each city name search a linear ~~arr~~ integer array for its label.

Algo:

→ label each city by a unique number from 1 to n. Call this Array C → For * line (X, Y) → Find labels for X & Y from array C → Then Union(C[X], C[Y]) → Report all Union find groups

Good: because of design simplicity & space

Bad because O(n) search for every line input

Problem II Grading Rules (50 points)

25

Design

(25pts) — Use either Option I or Option II

(~~less than~~ ≤ 15 points) — Any other option ~~that~~ that is worse than the ~~other~~ above two options

25

Implementation

(5) — Code readability

+

(20) = Does the code work? (testing)

If report is missing or is totally uninformative, Reduce 10 points for that