I & C SCI 46 Winter 2023
Lecture 5: Asymptotic Analysis I

2

Bipartite Checking

Algorithm to determine if G is bipartite

- G is undirected, connected.

BFS

odd: blue

even: gold

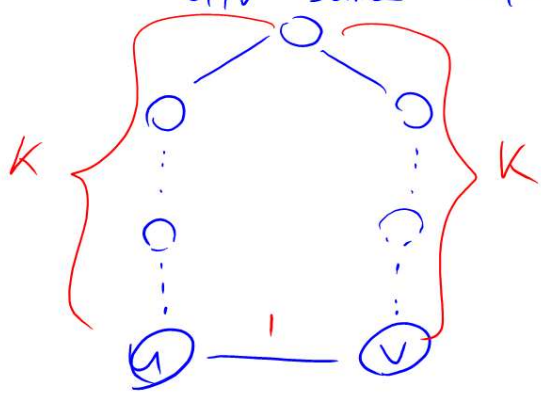
Claim: If G is bipartite, this demonstrates it

3 Bipartite Checking

If G **is not** bipartite, can we prove from algorithm?

find $e = (u, v)$ monochromatic

u, v same layer



4 Big Extra Credit Question

- **Input:** any simple graph G
- **Output:** chromatic number of G
- To claim extra credit:
  - Prove running time, polynomial in n and m
    $\mathcal{O}(n^{100000})$ okay
    $\mathcal{O}(2^n)$ is not.
  - Prove it is correct
- Worth an A+ in this class
  Actually worth more than that.

5

Running time of linear search

```
int linSearch(const std::vector<int> & numbers,
                    int target)
{
    int i;
    int n = numbers.size();
    for(i=0; i < n; i++)
        if( numbers[i] == target )
            return i;
    throw ElementNotFoundException("msg");
}
```

growth rate

worst? $O(n)$

5

Running time of linear search

```
int linSearch(const std::vector<int> & numbers,
                    int target)
{
    int i;
    int n = numbers.size();
    for(i=0; i < n; i++)
        if( numbers[i] == target )
            return i;
    throw ElementNotFoundException("msg");
}
```

- Does this time change if the vector is sorted?

6

Running time of binary search

```
int binarySearch(const std::vector<int> & numbers,
                 int target){
    return binarySearch(numbers, target, 0,
                 numbers.size() - 1);
}

int binarySearch(const std::vector<int> & numbers,
             int target, int low, int high)
{
if( low > high )
    throw ElementNotFoundException("msg");
int mid = (low + high) / 2;
if( numbers[mid] == target )
    return mid;
else if (target < numbers[mid])
    return binarySearch(numbers, target, low, mid-1);
else
    return binarySearch(numbers, target, mid+1, high);
}
```

7

What is binary search doing?

2	4	5	7	8	9	12	14	17	19	22	25	27	28	33

[1, 1000) ⟶ [501, 1000]

[751, 1000]

[751, 874]

[813, 874]

[813, 843]

[829, 843]

[829, 835] 832 !

8

Lower Bounds?

- Potential new solution to `find-min`
- Arbitrary input vector of size n
- Better than $\mathcal{O}(n)$ time.
- Believable? Why or why not?

9

Which is better? Why?

- Input: an array A of size n.
- Algorithm 1: $20n$ operations
- Algorithm 2: n^2 operations
- Which is better?