COURSE SELECTION PROGRAM

Kamden Ashmore, Garrett Kemper, Kishan Patel, Andrew Teh

MICROCOSM

- Intended for guidance counselors
- Generates potential schedules for large datasets of students
- Makes schedule creation efficient and automated

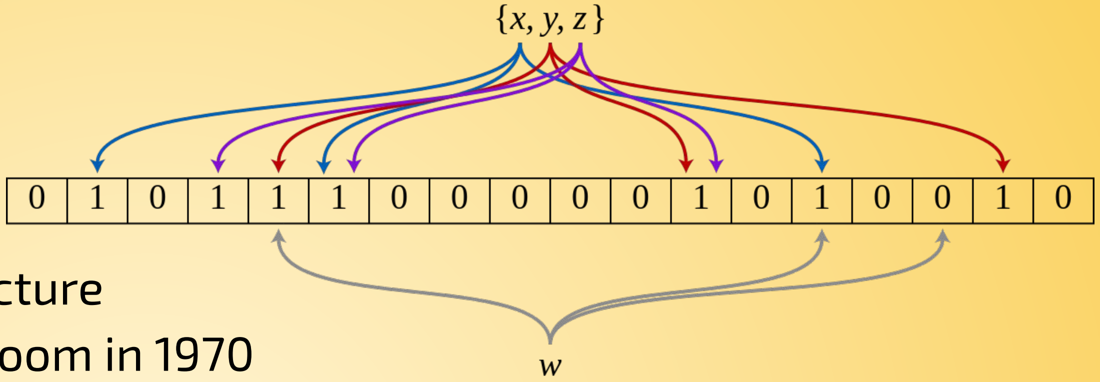
RANGE TREE

- First introduced by Jon Louis Bentley in 1979
- An ordered tree that efficiently reports all points within a given range and is typically used in two or higher dimensions
    - Efficiently solves multidimensional range queries
- With 1 dimension, it is essentially a balanced binary search tree
    - Ensures efficient search, insertion, and deletion operations with the time complexity being O (log n)
    - Time complexity of range query operation is O(log n)
- Range Trees can only be employed for range queries
    - Not efficient for operations such as nearest neighbor or exact match queries

RANGE TREE - OUR IMPLEMENTATION

- Used to retrieve classes within a specified range based on the capacity of the class.
- Range tree stores the percentage of how much the class is filled
  - Nodes contain percentage and name of class
  - Classes with less than or equal to the percentage go the the left while the higher ones go to the right
- Queried to find classes that are full or less than or equal to 30% full
- Seemed the easiest to comprehend when trying to find what classes are filled and what classes still have capacity

Bloom Filter

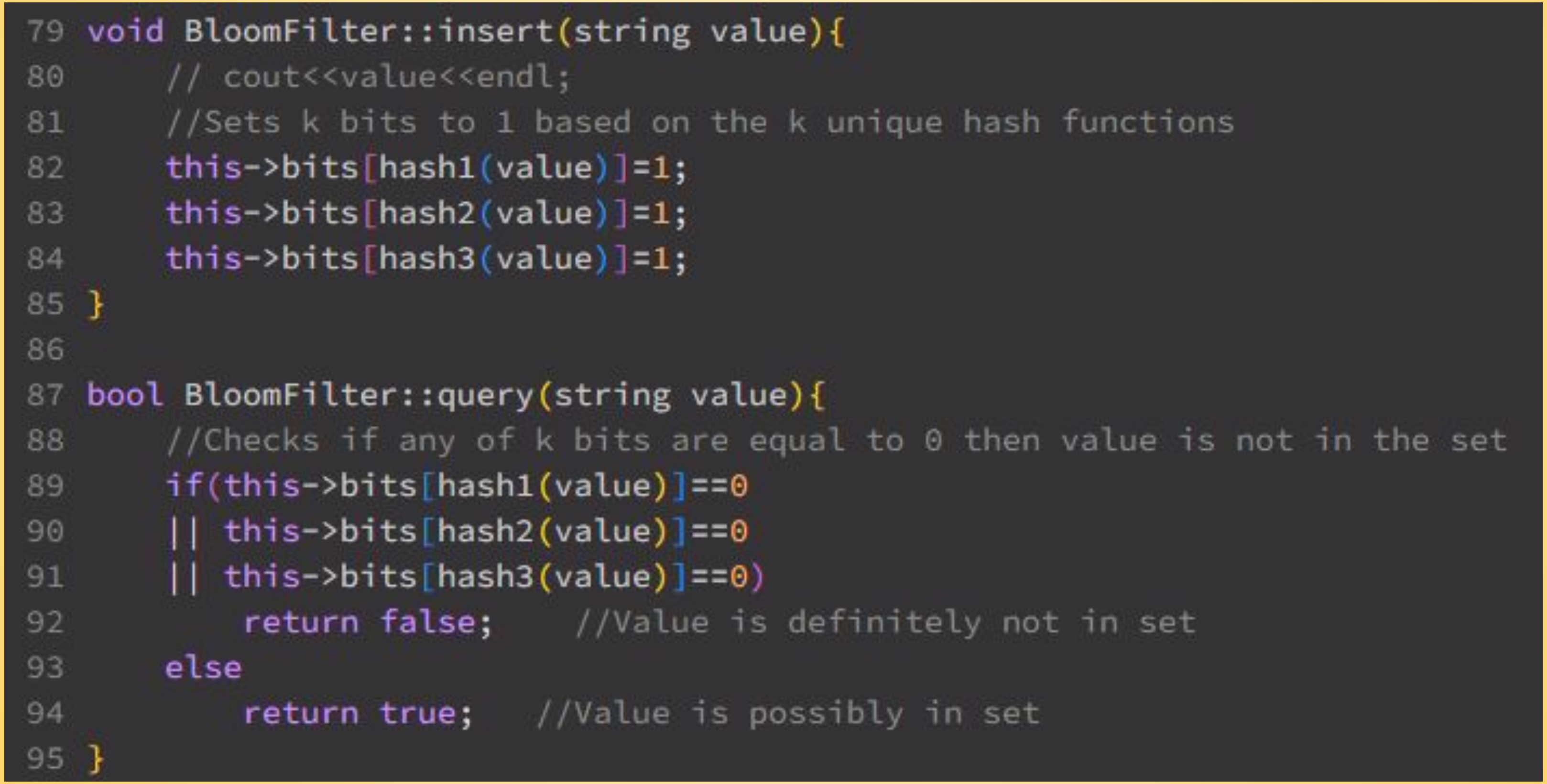


- Probabilistic based data structure
- Created by Burton Howard Bloom in 1970
- Allows to quickly test if an element is in a set
  - Possible to return false positives, impossible to return false negatives
  - “Possibly in set” or “Definitely not in set”
- Bit array of length m
  - All zeros when empty
- To insert, hash the element k times and set the indices returned to 1
- To query, use same hash functions and check if all indices are 1
  - If any are 0, the element is definitely not in set. If all are 1, it is possible it is in the set.
- Highly space efficient compared to trees, sets, or other similar structures
- Time Complexity: O(k)
- Space Complexity: O(m)

BLOOM FILTER - OUR IMPLEMENTATION

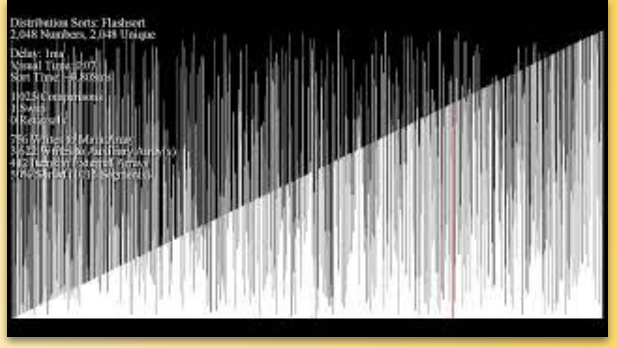
- Used to store and query the list of classes each student has previously taken
- Significantly more efficient than storing the actual classes or strings
  - A student could take over 50 classes
- Uses bit size of m=200 and
- Used to check for prerequisites and to avoid scheduling classes that have already been taking, therefore it is more important to know it is “Definitely not in set” than if it is in the set.
- Fast query and insertions are also important as the program needs to be efficient to schedule hundreds to thousands of students

BLOOM FILTER - CODE SNIPPET


```cpp
void BloomFilter::insert(string value){
    // cout<<value<<endl;
    //Sets k bits to 1 based on the k unique hash functions
    this->bits[hash1(value)]=1;
    this->bits[hash2(value)]=1;
    this->bits[hash3(value)]=1;
}

bool BloomFilter::query(string value){
    //Checks if any of k bits are equal to 0 then value is not in the set
    if(this->bits[hash1(value)]==0
    || this->bits[hash2(value)]==0
    || this->bits[hash3(value)]==0)
        return false;    //Value is definitely not in set
    else
        return true;   //Value is possibly in set
}
```

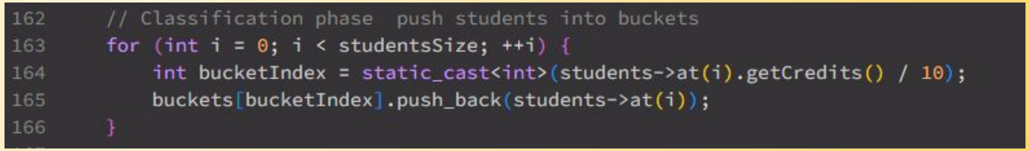
FLASH SORT



- created by Karl-Dietrich Neubert in 1998
- Flashsort partitions elements into buckets based on a hashing function or range calculation, similar to bucket sort.
- However, Flashsort specifically focuses on sorting elements within a specific range to take advantage of data characteristics like repetition and distribution.
- Works best on uniformly distributed data

FLASH SORT - OUR IMPLEMENTATION

- Used to sort students from least credits to most
  - More senior students get priority on classes
- Flashsort only works when the range of data is known
  - Credits range from 0 - 120
- Time complexity: O(n + m),
  - n is the number of elements
  - m is the number of distinct elements within the range
- Space Complexity: O(M)
  - M is number of buckets



```
    // Classification phase  push students into buckets
    for (int i = 0; i < studentsSize; ++i) {
        int bucketIndex = static_cast<int>(students->at(i).getCredits() / 10);
        buckets[bucketIndex].push_back(students->at(i));
    }
```

Code Snippet

SCOPE

- Scope needed to be limited to be reasonable for the timeframe
- Focused only on Computer Science BS curriculum
    - 23 classes, mostly based off of their actual schedules in Ecampus
    - 300 students, randomly generated
- Simplified schedules to either MWF or TTh
- CSV files for input/output
- Only “Must Take Before” prerequisites

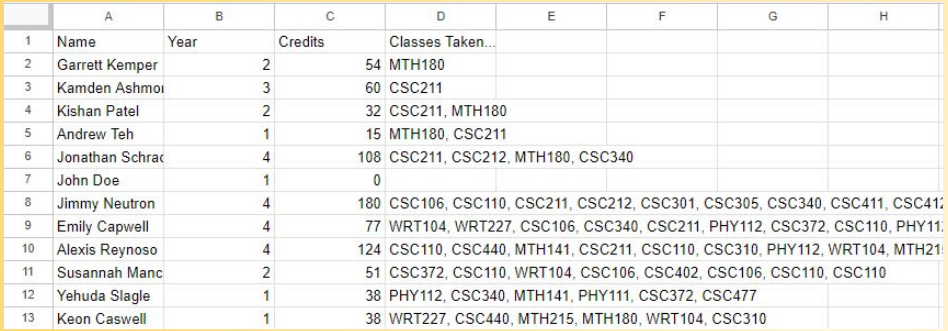
REQUIREMENTS

- Limited space in each class
- Cannot schedule overlapping classes to a student
- Maximum of 19 credits per student
- All prerequisites must have been taken and passed
- Output each student's classes to a CSV file
- Output "errors" to the console
    - Low enrollment classes, full classes, students with low credits (less than 12)

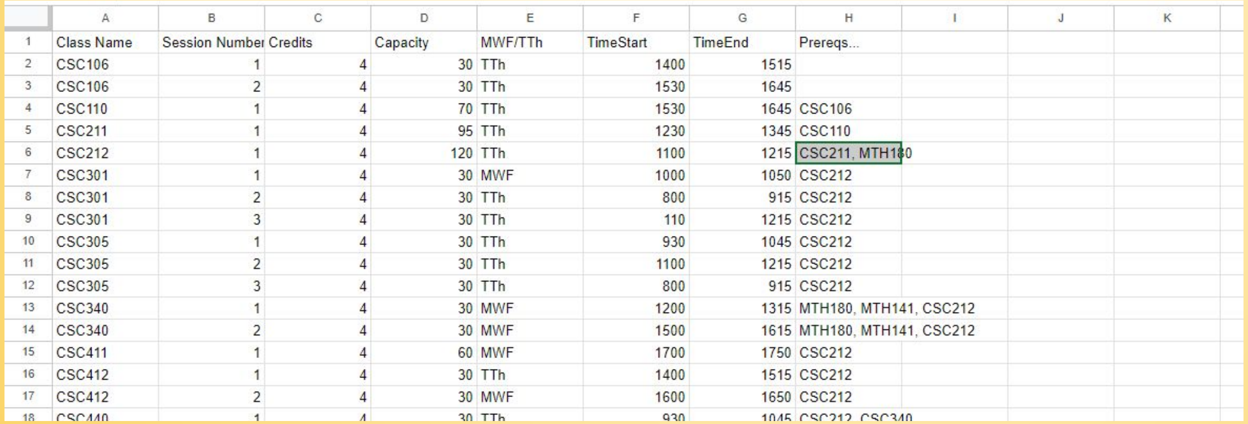
USAGE

- Provides a quick, efficient way to assign a large number of students to their necessary classes
- Provides feedback towards potential issues in the current class scheduling
- Allows for easy implementation with other systems as input and output files are all .CSV
- Prioritizes students with more credits as they are closer to graduation

DATA EXAMPLE

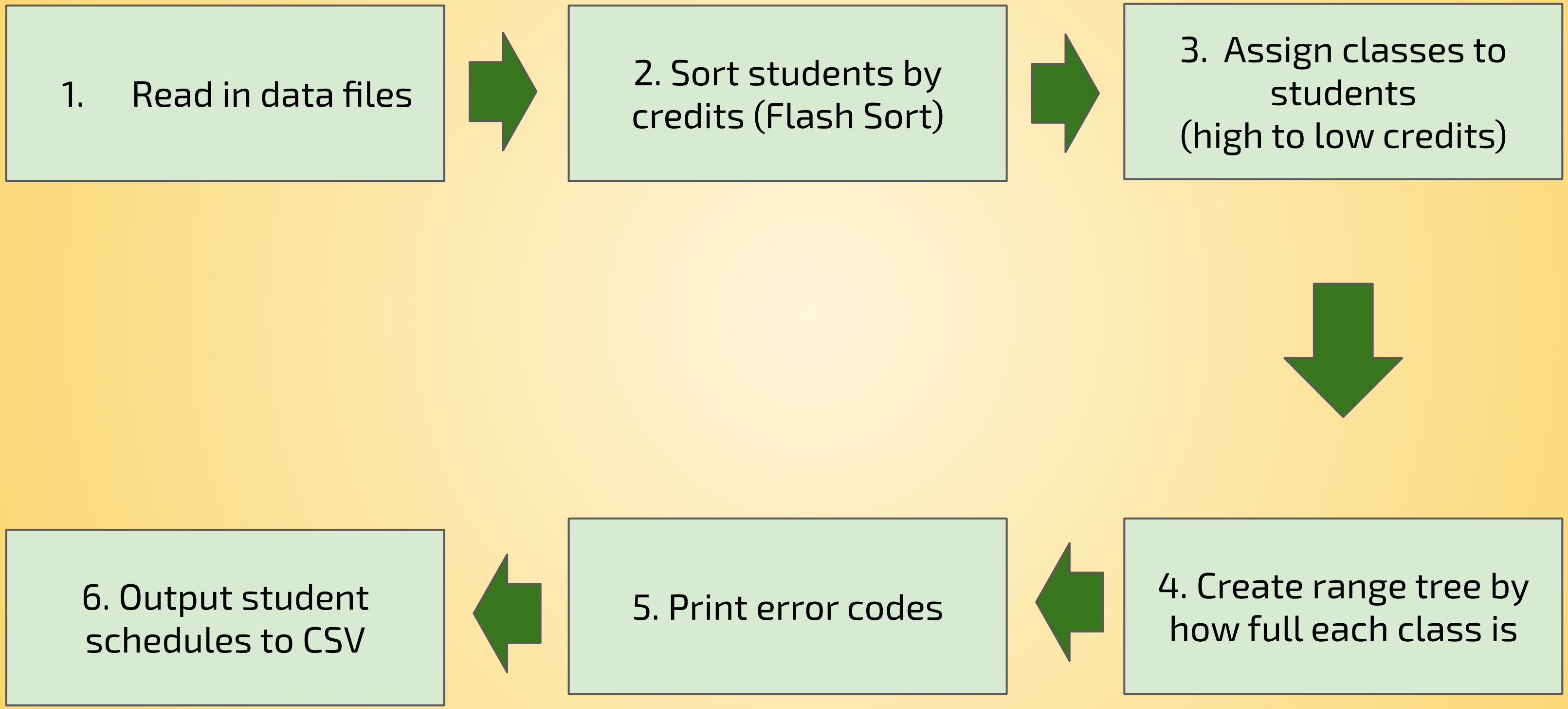


	A	B	C	D
1	Name	Year	Credits	Classes Taken...
2	Garrett Kemper	2	54	MTH180
3	Kamden Ashmoı	3	60	CSC211
4	Kishan Patel	2	32	CSC211, MTH180
5	Andrew Teh	1	15	MTH180, CSC211
6	Jonathan Schrac	4	108	CSC211, CSC212, MTH180, CSC340
7	John Doe	1	0	
8	Jimmy Neutron	4	180	CSC106, CSC110, CSC211, CSC212, CSC301, CSC305, CSC340, CSC411, CSC41
9	Emily Capwell	4	77	WRT104, WRT227, CSC106, CSC340, CSC211, PHY112, CSC372, CSC110, PHY11
10	Alexis Reynoso	4	124	CSC110, CSC440, MTH141, CSC211, CSC110, CSC310, PHY112, WRT104, MTH21
11	Susannah Manc	2	51	CSC372, CSC110, WRT104, CSC106, CSC402, CSC106, CSC110, CSC110
12	Yehuda Slagle	1	38	PHY112, CSC340, MTH141, PHY111, CSC372, CSC477
13	Keon Caswell	1	38	WRT227, CSC440, MTH215, MTH180, WRT104, CSC310



	A	B	C	D	E	F	G	H
1	Class Name	Session Number	Credits	Capacity	MWF/TTh	TimeStart	TimeEnd	Prereqs...
2	CSC106	1	4	30	TTh	1400	1515	
3	CSC106	2	4	30	TTh	1530	1645	
4	CSC110	1	4	70	TTh	1530	1645	CSC106
5	CSC211	1	4	95	TTh	1230	1345	CSC110
6	CSC212	1	4	120	TTh	1100	1215	CSC211, MTH180
7	CSC301	1	4	30	MWF	1000	1050	CSC212
8	CSC301	2	4	30	TTh	800	915	CSC212
9	CSC301	3	4	30	TTh	110	1215	CSC212
10	CSC305	1	4	30	TTh	930	1045	CSC212
11	CSC305	2	4	30	TTh	1100	1215	CSC212
12	CSC305	3	4	30	TTh	800	915	CSC212
13	CSC340	1	4	30	MWF	1200	1315	MTH180, MTH141, CSC212
14	CSC340	2	4	30	MWF	1500	1615	MTH180, MTH141, CSC212
15	CSC411	1	4	60	MWF	1700	1750	CSC212
16	CSC412	1	4	30	TTh	1400	1515	CSC212
17	CSC412	2	4	30	MWF	1600	1650	CSC212
18	CSC440	1	4	30	TTh	930	1045	CSC212, CSC340

FUNCTION


1. Read in data files
2. Sort students by credits (Flash Sort)
3. Assign classes to students (high to low credits)
4. Create range tree by how full each class is
5. Print error codes
6. Output student schedules to CSV

DEMONSTRATION

PROS AND CONS

Pros:

- Very efficient
- Can support very large amounts of data
- Output is clear and concise
- Easily marketable if we were to expand the scope

Cons:

- Doesn't take into account different workload desires
- Classes that require concurrent enrollment aren't accounted for
- Does not support non MWF/TTh schedules

PROBLEMS ENCOUNTERED

- Flash Sort originally used 4 buckets
  - Increased to 100 for efficiency
- Classes enrolled stored in a Bloom Filter
  - Lost data such as session number
  - False positives
  - Less need for storage space
  - This was changed to a vector of classes
- The Range Tree querying for classes student can take
  - Prerequisites did not fit well into a tree structure
  - Less efficient than just checking each class one by one
  - Instead, used to find classes based on their enrollment percentages

FUTURE

- Adding a GUI
- A toggleable graphic for individual students to see their schedules in a weekly calendar format
- Expanding the scope to include multiple schools/majors

THANK YOU

Any Questions?