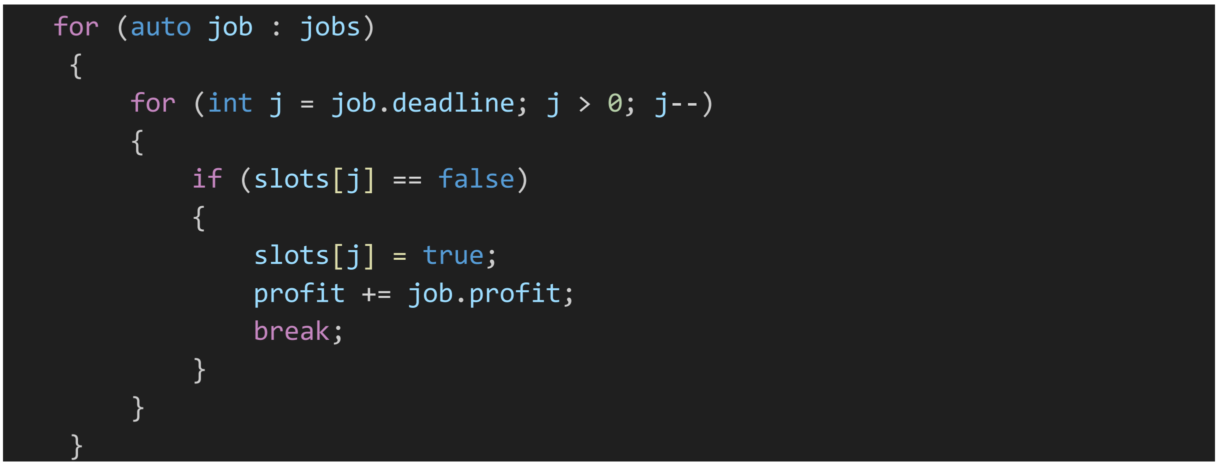
AZA – assignment

Adrian Maslak

1. JOB DEADLINE SCHEDULING PROBLEM

Without including sorting time complexity we can calculate BIG-0 as O(n^2) where n represents number of jobs.



```
for (auto job : jobs)
 {
     for (int j = job.deadline; j > 0; j--)
     {
         if (slots[j] == false)
         {
             slots[j] = true;
             profit += job.profit;
             break;
         }
     }
 }
```

Based on the provided code snippet we can analyze that it is indeed true that bigO is n^2 because have 2 for loops which in coding means that the notation is exponential

2. DEADLINE N PROBLEM USING DISJOINT SET

Without including sorting into time complexity we can calculate BIG-O as O(N*log(maxDeadline)) where *n* represents number of jobs.

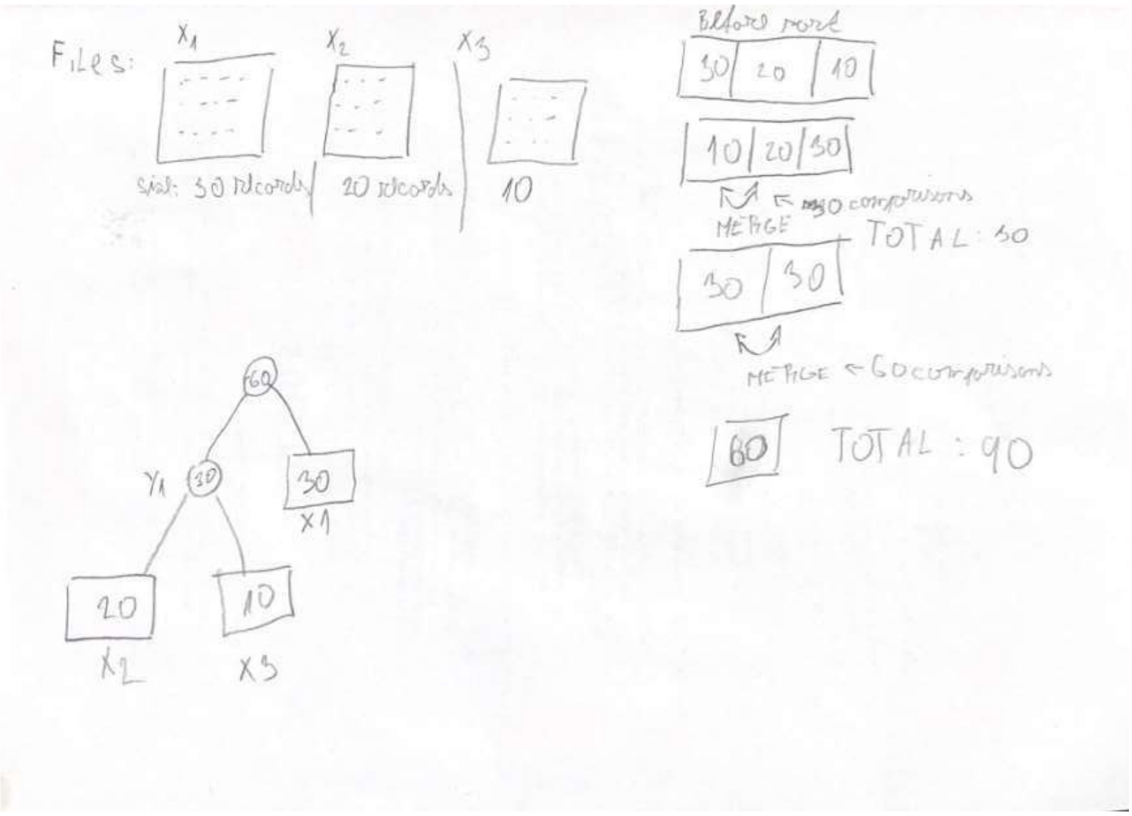
Since the program iterates through the jobs *n* times, finding *maxdeadline* is found by iterating through the data set *n times* while initializing disjoint data set iterates *maxDeadline* times.

3. Merge N files using GREEDY approach

We can describe Greedy approach as choosing the local optimum by choosing 2 smallest options that are right now available without thinking about the future.

A. Show two different inputs for your implementation and analyse each step of your algorithm.

```
Enter the number of files: 3
Enter the sizes of the files:
Size of file 1: 30
Size of file 2: 20
Size of file 3: 10
Merge Steps:
---------------------------
Merged 10 and 20 into 30 (Total Record Moves: 30)
Merged 30 and 30 into 60 (Total Record Moves: 90)
---------------------------
Total Record Moves: 90
```

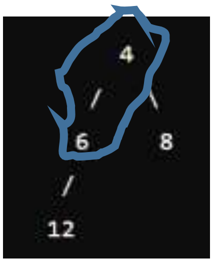


Input 2:

8,4,6,12

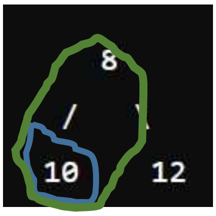
Initialization –

Representation of min heap

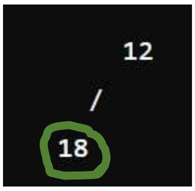


```
     4
    / \
   6   8
  /
 12
```

Action : Merging 2 smalleset



```
      8
     / \
   10   12
```



```
      12
     /
   18
```

Merge Steps:

Merged 4 and 6 into 10 (Total Record Moves: 10)

Merged 8 and 10 into 18 (Total Record Moves: 28)

Merged 12 and 18 into 30 (Total Record Moves: 58)

Total Record Moves: 58

B: Analyse your algorithm and show the results using order notation.

```
void buildMinHeap(vector<int> &heap)
{     int size =
heap.size();
    // Start from the last non-leaf node and heapify each
node      for (int i = size / 2 - 1; i >= 0; i--)
heapify(heap, size, i);
}
```

Time complexity O(n)

Merge operations:

Number of Merge Steps: n−1 (for n files)

Extracting the Two Smallest Files: O(logn)

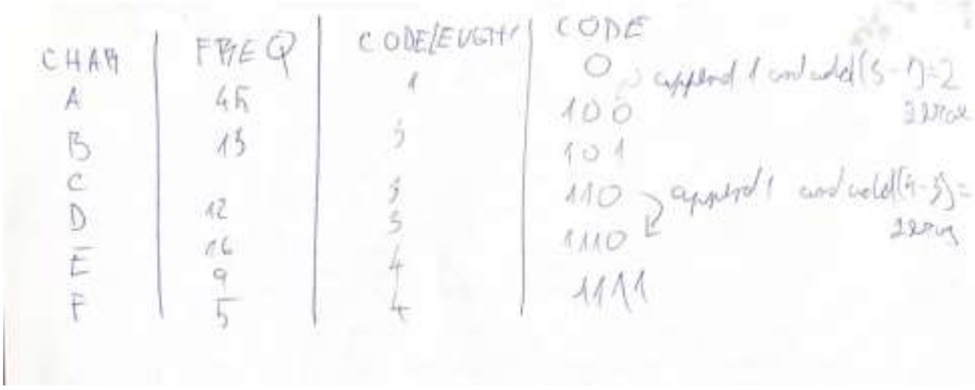
Inserting the Merged File: O(logn)

Time complexity per merge O(2logn)=O(logn)

Time complexity for all merge steps O((n−1)logn)=O(nlogn)

4. HUFFMAN

Canonical coding



CHAR	FREQ	CODELENGTH	CODE
A	45	1	0
B	13	3	100
C	12	3	101
D	16	3	110
E	9	4	1110
F	5	4	1111

append 1 and add (3-1)=2 zeros

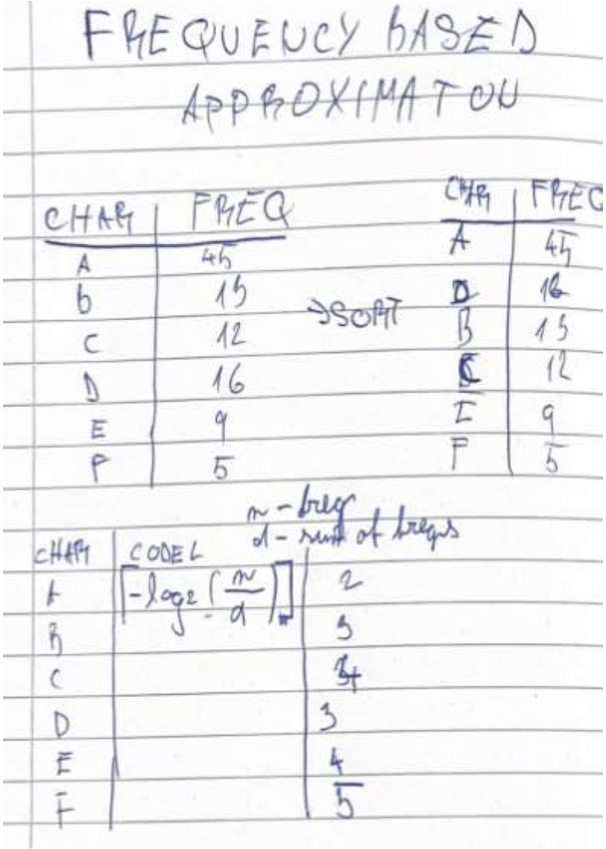
append 1 and add (4-3)= 1 zero

Sort characters by **frequency** (ascending order). If frequencies are the same, sort **alphabetically**. Assign shorter code lengths to higher-frequency characters, ensuring:

- Lower frequency → Longer code length
- Higher frequency → Shorter code length
- Increment the binary number for each character.
- For a new code length group, **pad with zeroes on the right** to match the required length.

If we exclude Bubble sort from the code bigO(n^2) then we can analyse that canonical coding notation is bigO(n) since we are only iterating through the symbols, bit shift operation don't have almost any weight in this notation

Frequency Based Approximation



FREQUENCY BASED APPROXIMATON

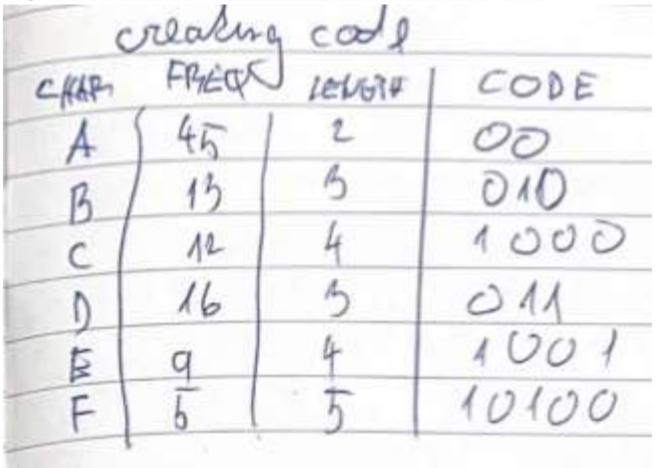
CHAR	FREQ
A	45
b	15
C	12
D	16
E	9
F	5

→SORT

CHAR	FREQ
A	45
D	16
B	15
C	12
E	9
F	5

n - freq
d - sum of freqs

CHAR	CODE L	
A	$\lceil -\log_2 (\frac{n}{d}) \rceil$	2
B		3
C		4
D		3
E		4
F		5



creating code

CHAR	FREQ	LENGTH	CODE
A	45	2	00
B	15	3	010
C	12	4	1000
D	16	3	011
E	9	4	1001
F	5	5	10100

Example of decoding, since we have the table ready we can create sentecne 00011001000

We start by reading bits subsequently because no code can be prefix of other code

ADAC

That’s just a brief example of how the coding and decoding works.

Same as in the canonical coding the biO(n) because we are iterating through the symbols