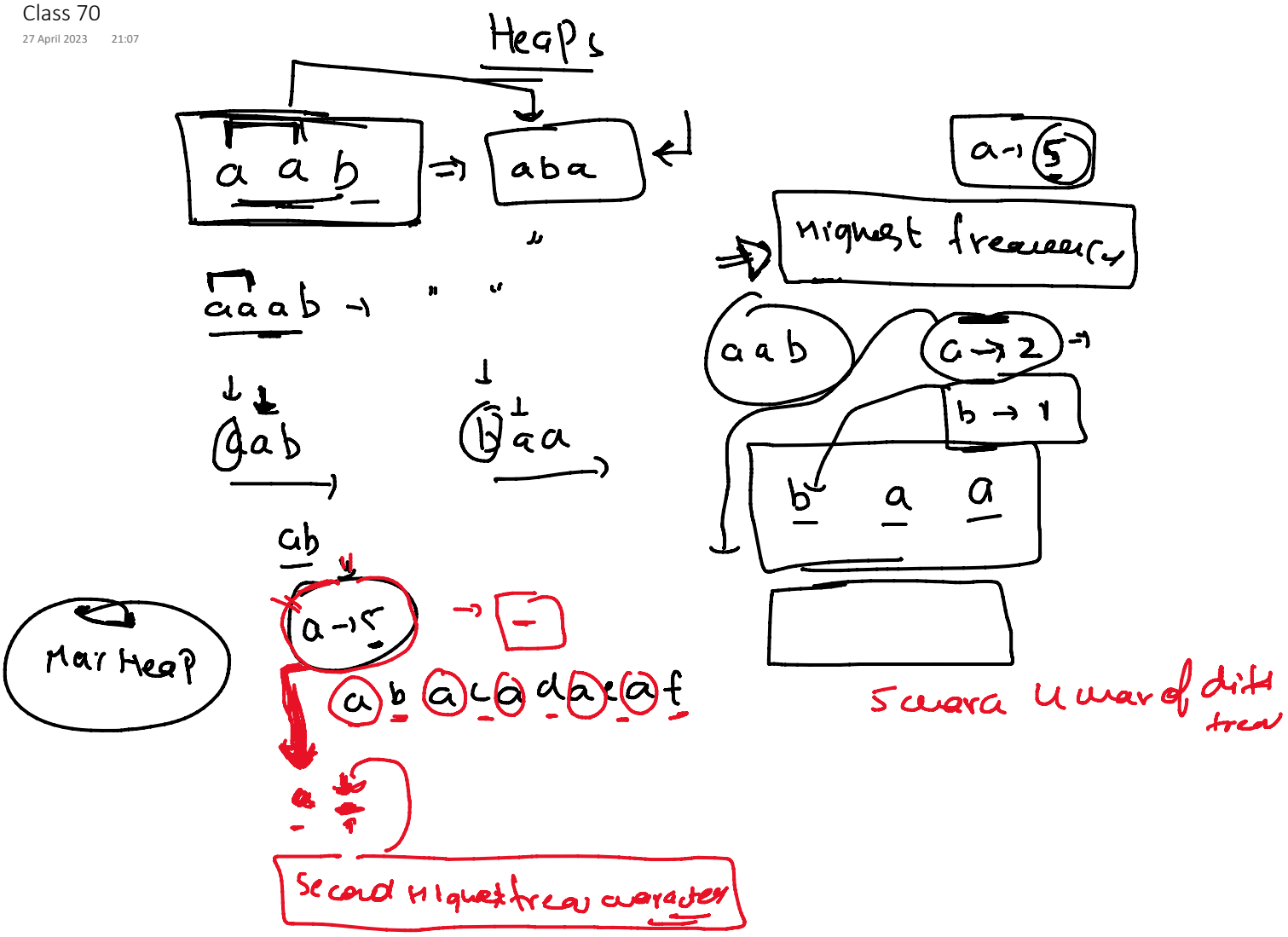
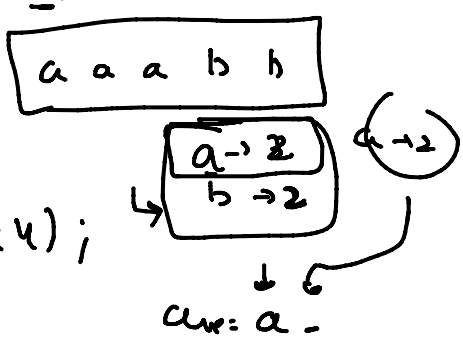
Class 70

27 April 2023 21:07

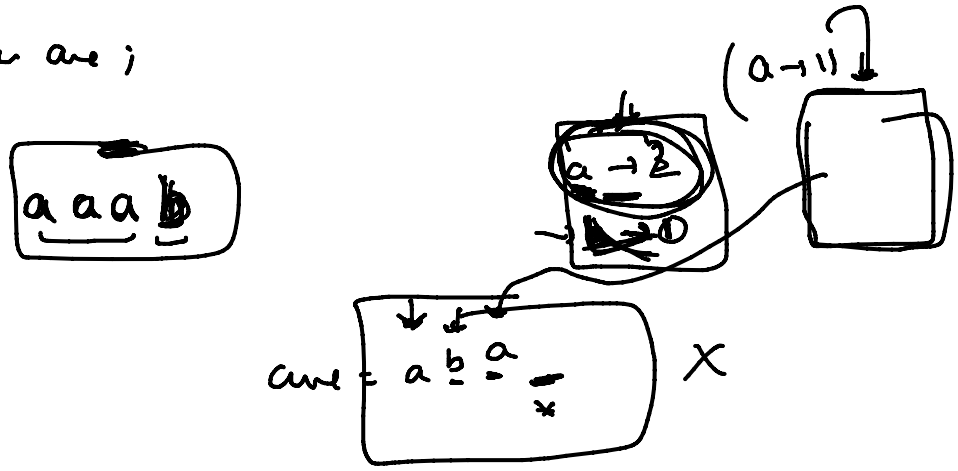


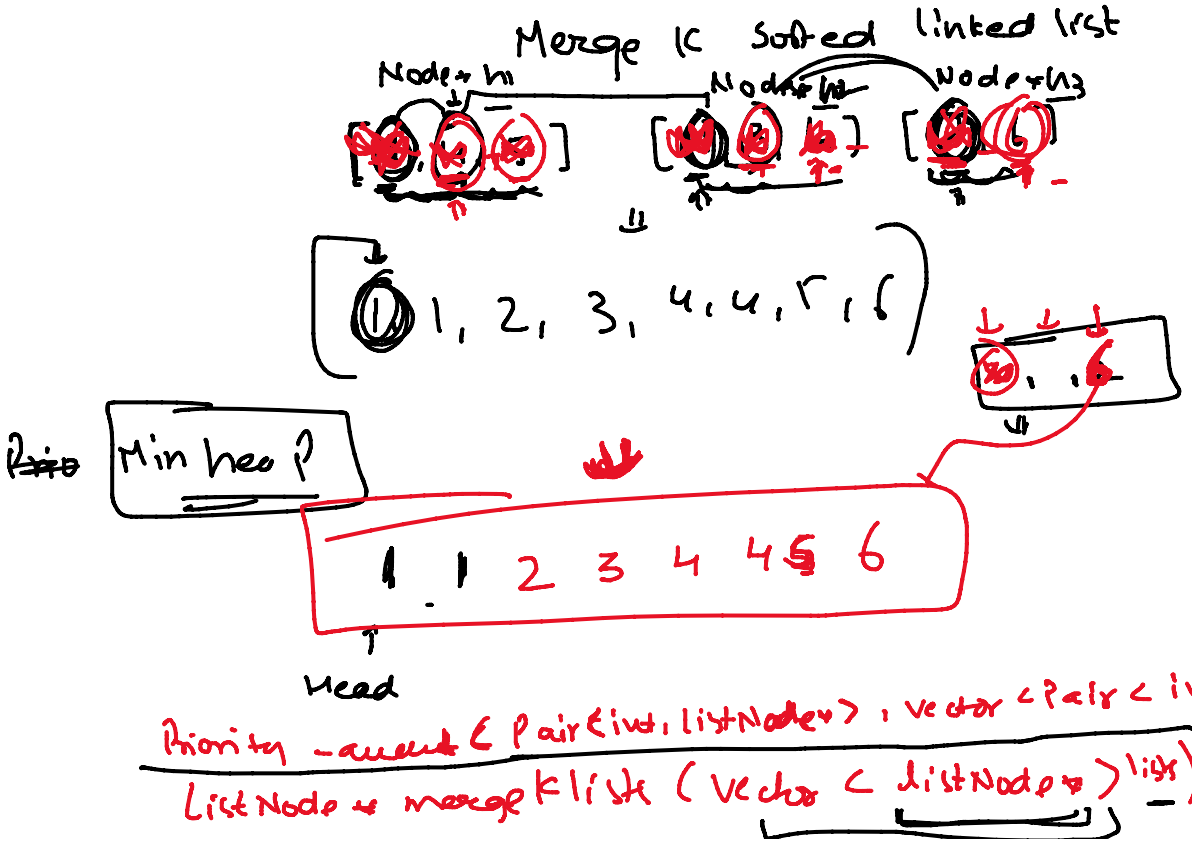

```
String reorganizeString ( string s) {
    int i, j, k, n;
    n = s.size();
    string ans = " ";
    Priority_queue < Pair < int , char >> Pq;
    map < char, int > ma;
    For ( i=0; i<n; i++ )
    { ma[ s[i] ]++;
    }
    For (auto i : ma )
    { Pq.push ({ i.second , i.first });
```

```
for (auto i : ma)
{ pq.push ({i.second, i.first});
}

auto cr = pq.top(); {first, second}
pq.pop();

ans += cr.second;

cr.first--;

if( (r.first != 0)
  pq.push({cur.first, cur.second});

for ( i=1; i<n; i++)

{ (cur) = pq.top();

  pq.pop();

  if ( (cur.second == ans[i-1])

  { if(pq.empty())
        return " ";

    auto (cr2) = pq.top();
    pq.pop();

    ans += cr2.second;

    cr2.first--;

    if ( (cur2.first != 0)
       pq.push({cur2.first, cur2.second});
  }

  else
```



```
else
{   ans += (r.second);
    r.first--;
}
if((r.first != 0)
   Pq.Push({r.first, r.second});
}
return ans;
```

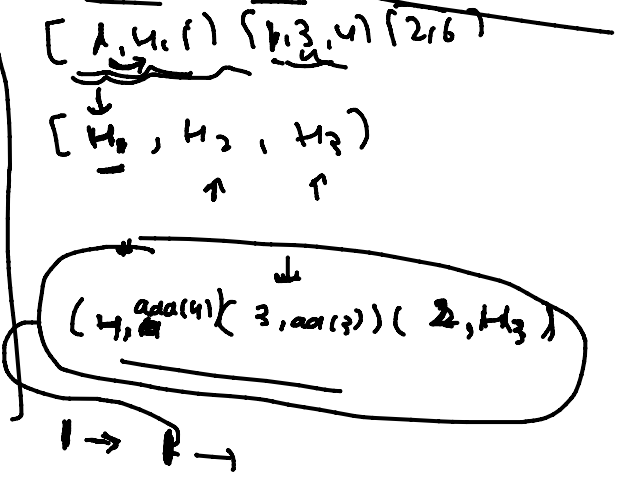




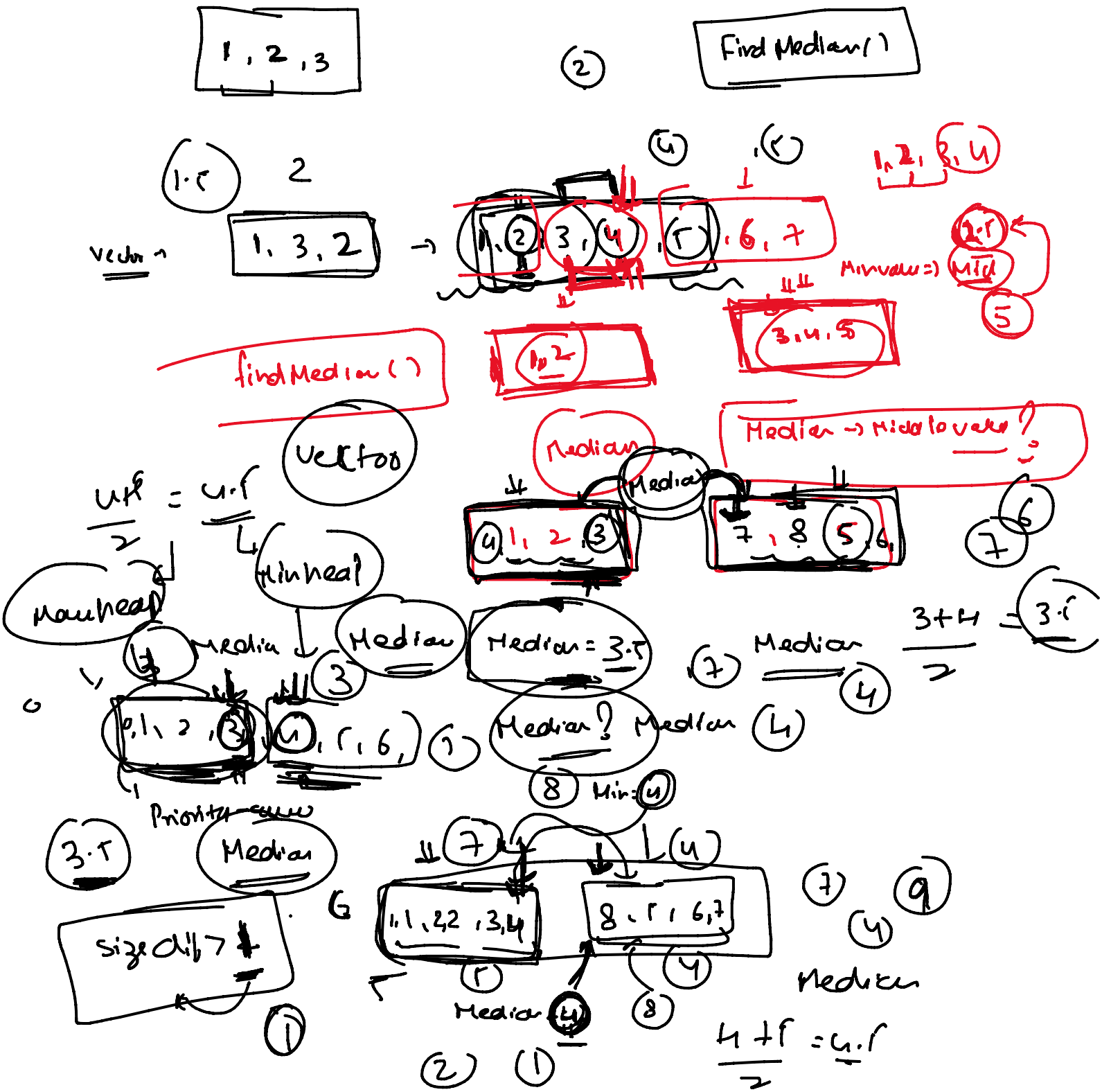
Priority_queue<pair<int, ListNode*>, vector<pair<int, ListNode*>>, greater<pair<int, ListNode*>>>

ListNode* mergeKlists(vector<ListNode*>& lists)

```
ListNode* mergeKlist(vector<listNode*>lists)
{
    For (auto i : lists)
    {  if (i != NULL)
       {  Pq.Push({i->val, i});
       }
    }
    ListNode* merge()
    {   if (Pq.size() == 0)
            return NULL;
        auto node = Pq.top().second;
        auto nodeval = Pq.top().first;
        Pq.pop();
        if (node->next)
        {  Pq.Push({node->next->val, node->next});
        }
        node->next = merge();
        return node;
    }
```



Median of Running Stream




```
Priority_queue <int> mxh;
Priority_queue <int, vector <int>, greater <int>> mnh;
double last = 0;
```

```
void add ( int num)
{   if ( mx.size() == mn.size() )
    {  if ( num >= last )
        {  mn.push ( num);
           last = mn.top();
        }
       else
        {  mx.push (num);
           last = mx.top();
        }
    }
   else if ( mx.size() > mn.size() )
    {   if (num <= last)
        {  int tp = mx.top();
           mx.pop();
           mn.push ( tp);
           mx.push (num);
           last = (mx.top() + mn.top()) / 2;
        }
       else
        {  mn.push(num);
           last = (mx.top() + mn.top()) / 2;
        }
    }
```

```
else if (mn.size() > mx.size())
{   if (num >= last)
    {   int tp = mn.top();
        mn.pop();
        mx.push(tp);
        mn.push(num);
        last = (mx.top() + mn.top()) / 2.0;
    }
    else
    {   mx.push(num);
        last = (mx.top() + mn.top()) / 2.0;
    }
}
```