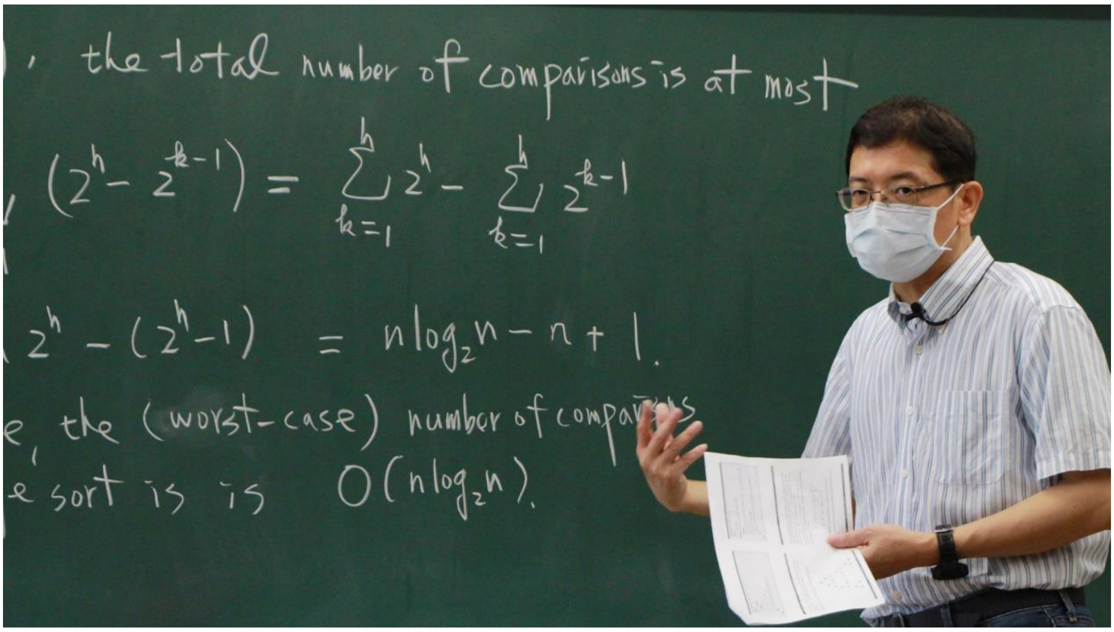
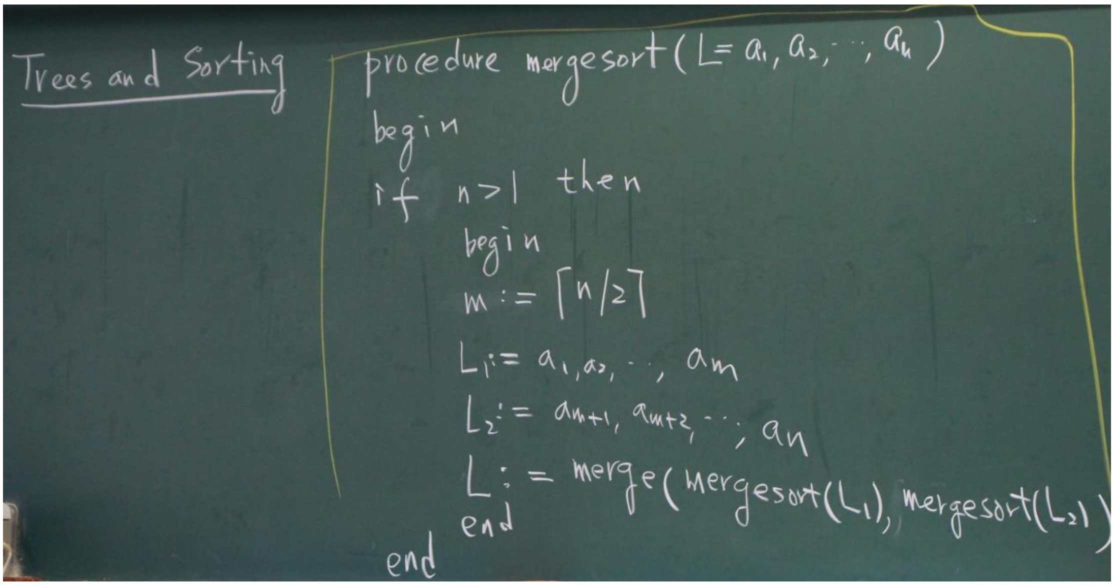
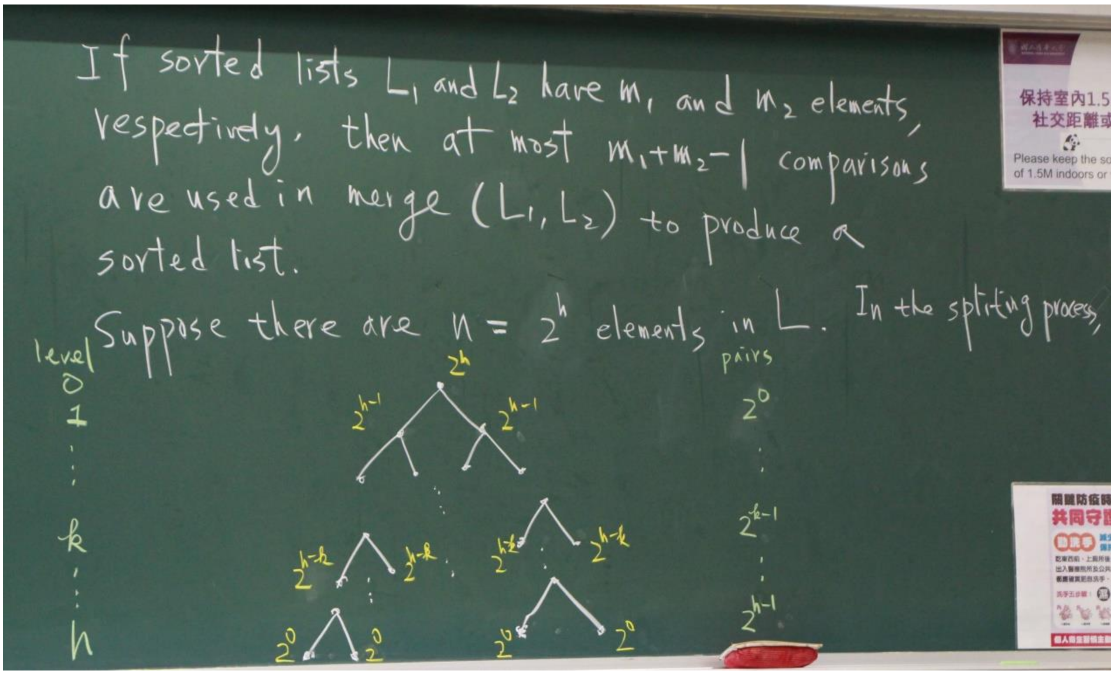
【10920 趙啟超教授離散數學 / 第 26 堂版書】

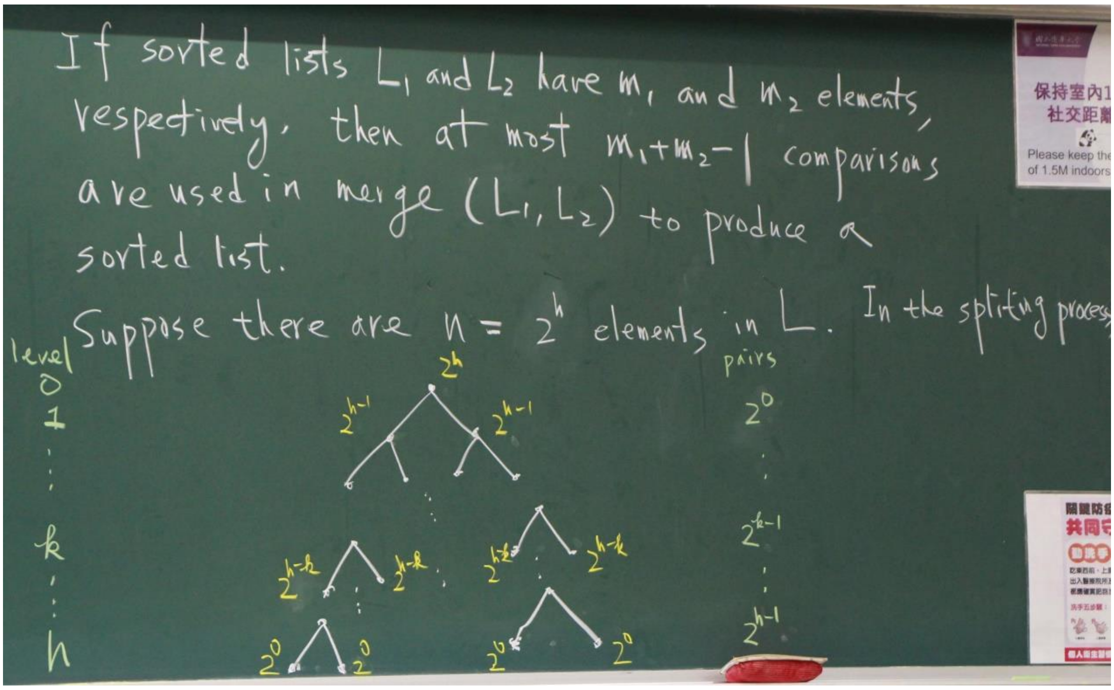




```
procedure merge (L1, L2)
begin
L: = empty list
while L1 and L2 are both nonempty
    begin
    romove smaller of the first elements
    of L1 and L2 and put it at the right end of L
   if one list is empty then
        romove all elements of the other list and append them to L
   end
end
```


```
procedure merge (L1, L2)
begin
L: = empty list
while L1 and L2 are both nonempty
    begin
    romove smaller of the first elements
    of L1 and L2 and put it at the right end of L
   if one list is empty then
        romove all elements of the other list and append them to L
   end
end
```


If sorted lists L_1 and L_2 have m_1 and m_2 elements, respectively, then at most m_1+m_2-1 comparisons are used in merge(L_1, L_2) to produce a sorted list.
Suppose there are $n = 2^h$ elements in L. In the spliting process,
level
0
1
k
h
pairs
2^0
2^{k-1}
2^{h-1}
2^h
2^{h-1}
2^{h-1}
2^{h-k}
2^0


If sorted lists L_1 and L_2 have m_1 and m_2 elements, respectively, then at most m_1+m_2-1 comparisons are used in merge(L_1, L_2) to produce a sorted list.
Suppose there are $n = 2^h$ elements in L. In the spliting process,
level
0
1
k
h
pairs
2^0
2^{k-1}
2^{h-1}
2^h
2^{h-1}
2^{h-1}
2^{h-k}
2^0

In the merging process, at level h, there are 2^{h-1} pairs of vertices. For each pair, there are two sublists each of size 2^0 and to merge the two sublists, at most $2^0+2^0-1 = 2^1-1$ comparisons is needed

Hence at level h at most $2^{h-1}(2^1-1) = 2^h - 2^{h-1}$ comparisons are used.

In general, at level k, $1 \le k \le h$, at most $2^{k-1}(2^{h-k}+2^{h-k}-1)$
$= 2^{k-1}(2^{h-k+1}-1) = 2^h - 2^{k-1}$ comparisons are used

In the merging process, at level h, there are 2^{h-1} pairs of vertices. For each pair, there are two sublists each of size 2^0 and to merge the two sublists, at most $2^0+2^0-1 = 2^1-1$ comparisons is needed

Hence at level h at most $2^{h-1}(2^1-1) = 2^h - 2^{h-1}$ comparisons are used.

In general, at level k, $1 \le k \le h$, at most $2^{k-1}(2^{h-k}+2^{h-k}-1)$
$= 2^{k-1}(2^{h-k+1}-1) = 2^h - 2^{k-1}$ comparisons are used

Consequently, the total number of comparisons is at most

$$\sum_{k=1}^{h} (2^h - 2^{k-1}) = \sum_{k=1}^{h} 2^h - \sum_{k=1}^{h} 2^{k-1}$$

$$= h2^h - (2^h - 1) = n\log_2 n - n + 1.$$

Therefore, the (worst-case) number of comparisons in merge sort is is $O(n\log_2 n)$.

Recall that the numbers of comparisons in bubble sort and insertion sort are both $O(n^2)$.

Consequently, the total number of comparisons is at most

$$\sum_{k=1}^{h} (2^h - 2^{k-1}) = \sum_{k=1}^{h} 2^h - \sum_{k=1}^{h} 2^{k-1}$$

$$= h\,2^h - (2^h - 1) = n\log_2 n - n + 1.$$

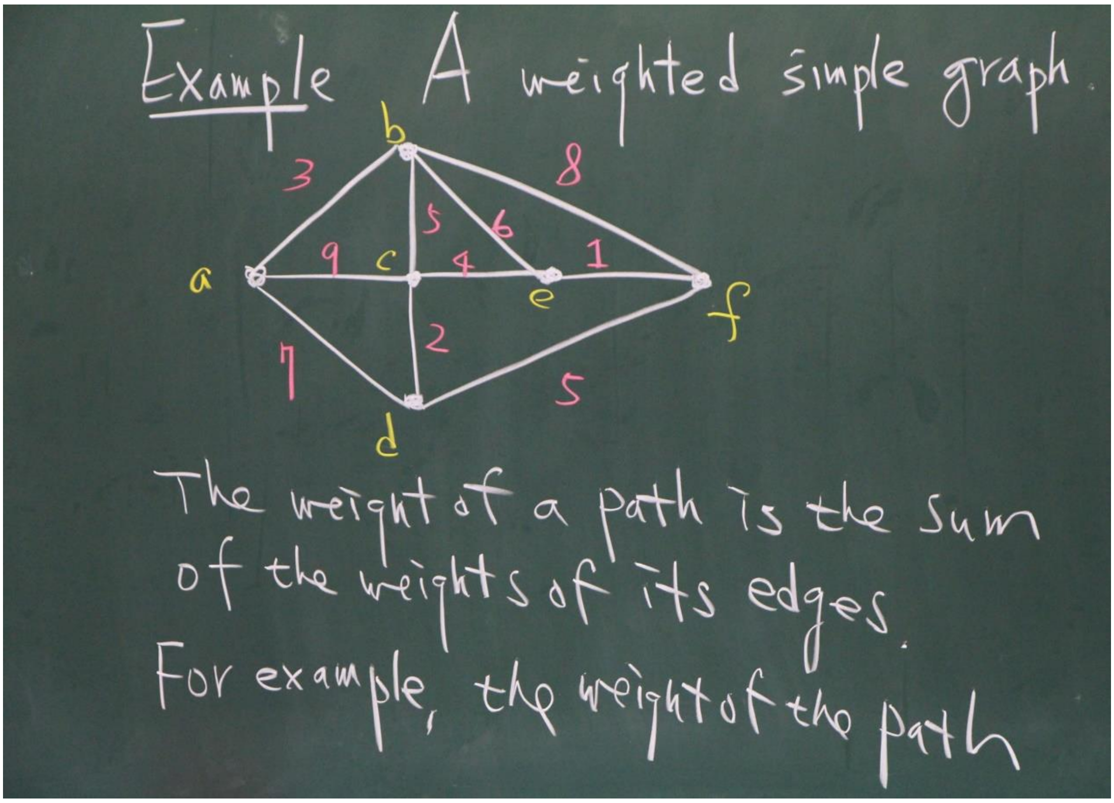
Therefore, the (worst-case) number of comparisons in merge sort is is $O(n\log_2 n)$.

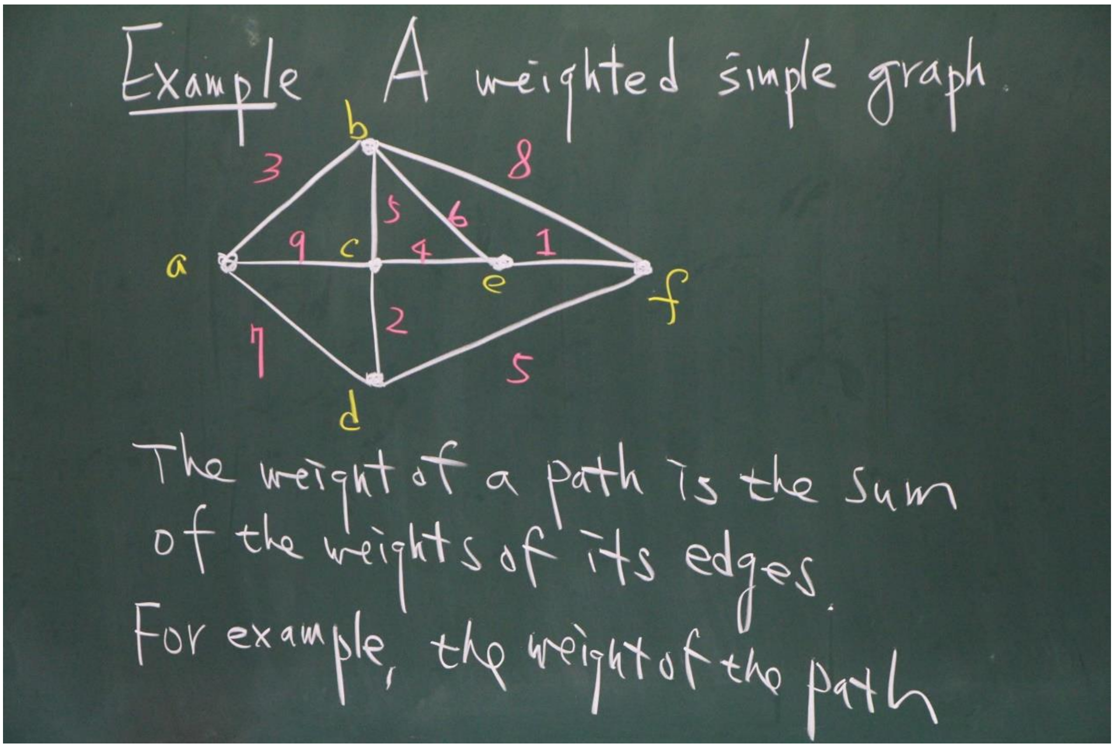
Recall that the numbers of comparisons in bubble sort and insertion sort are both $O(n^2)$.

Optimization and Matching

Shortest-Path Problem

Consider an undirected connected simple graph $G=(V,E)$. There is a weight $w(u,v)>0$ for each edge $\{u,v\}\in E$. If $\{u,v\}\notin E$, then $w(u,v)=\infty$.


Example A weighted simple graph.
b
3
8
5
6
a
9
c
4
1
e
f
2
7
5
d
The weight of a path is the sum of the weights of its edges.
For example, the weight of the path

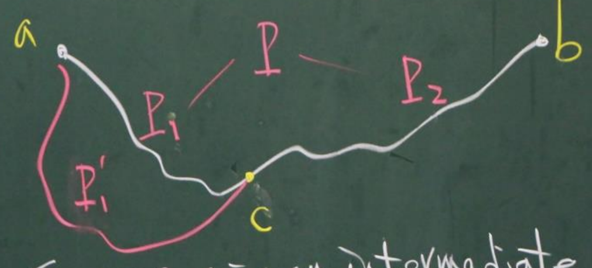

Example A weighted simple graph.
b
3
8
5
6
a
9
c
4
1
e
f
2
7
5
d
The weight of a path is the sum of the weights of its edges.
For example, the weight of the path

$a\,d\,c\,b\,f$ is $w(a,d)+w(d,c)+w(c,b)+w(b,f)=7+2+5+8=22$.

We would like to find the path from some vertex to some other vertex with the least weight.

$\Rightarrow$ the shortest-path problem.

Principle of optimality:



Suppose c is an intermediate vertex along the shortest path P from a to b. The the portion of the path from a to c, P_1, must be the shortest path among all paths from a to c. Otherwise, if there were a shorter path P_1' from a to c, then the path $P_1' P_2$ would be shorter than P.