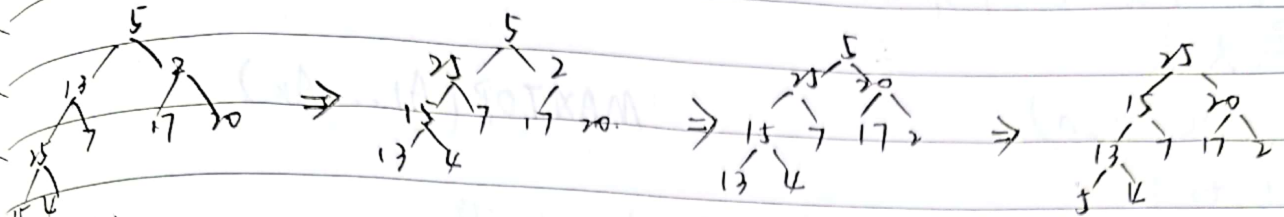
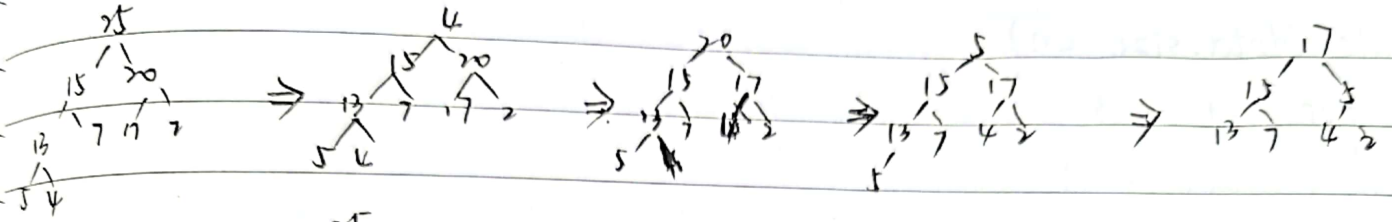
PS4

1. (a)

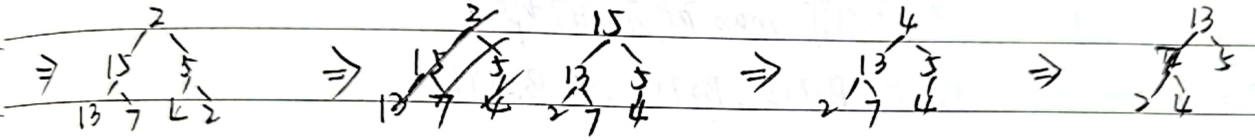


(b)



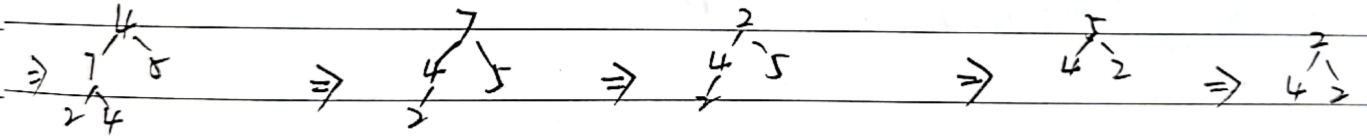
25

25 20



25 20 17

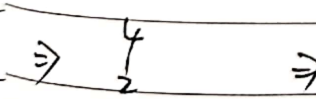
25 20 17 15



25 20 17 15 13

25 20 17 15 13 7

25 20 17 15 13 7 5



25 20 17 13 7 5 4 2

2. 把 k 个已排好的 list 排成最大堆

每次比较并取出 k 个堆顶元素中最大的，而后堆重排

用最大堆

```
SORT(A1 ... Ak, k, n)
  for i=1 to k
    Bi=BUILD-MAX-HEAP(Ai)  // 新建 k 个堆
  int count =0
  int data[n];
  while (count <n)

    int index              // 存储 max 所在的堆
    Ccount=BUILD-MAX-HEAP(B1[1], B2[1] ... Bk[1])
    data[count] = Ccount[1]
    count = count +1
    MAXHeapify(Bindex)
  return data
```

Time：每次通过新建一个最大堆取 max：$\lg k$

共 n 次 $\Rightarrow O(n \lg k)$

3.

(a) 归纳：$n=2$ 时显然正确

假设 $n \le 2^k$ 正确（n是2的幂）

当 $n=2^{k+1}$ 时

Cruel $(A[1 \ldots 2^{k+1}])$ 首先调用了 Cruel $(A[1 \ldots 2^k])$

& Cruel $(A[2^k+1 \ldots 2^{k+1}])$

由归纳假设，此时 $A[1 \ldots 2^k]$ & $A[2^k+1 \ldots 2^{k+1}]$ 均已排好

且 Unusual 有能力将2个长度为 2^{k-1} 的已排合并为长度 2^k 已排

此时 Unusual $(A[1 \ldots 2^{k+1}])$ 已进入 3个递归

- $A[1 \ldots 2^k]$
- $A[2^k+1 \ldots 2^{k+1}]$
- $A[2^{k-1} \ldots 3 \cdot 2^{k-1}+1]$

注意：由归纳假设，这3个递归均可成功排序

即因为它们都由2个已排好的长度为 2^{k-1} 的部分拼接而成

即 Unusual $(A[1 \ldots 2^{k+1}])$ 排序成功

由归纳假设：$\forall k$，$\forall n$ 成立

(b) 反例：不妨令 $n=4$，$A=[3,4,1,2]$

则 Unusual $(A[1 \ldots 4])$ 的变化为：

3, 4, 1, 2

↓ Unusual $(A[1 \ldots 2])$

3, 4, 1, 2

↓ Unusual $(A[3 \ldots 4])$

3, 4, 1, 2

↓ Unusual $(A[2 \ldots 3])$

3, 1, 4, 2

失败!!

(c) 反例: $n=4$, $A=[4,3,2,1]$

则调用 Unusual 进行变化为:

$4.3.2.1 \longrightarrow 4.2.3.1 \longrightarrow 2.4.3.1 \longrightarrow 2.3.4.1 \longrightarrow 2.3.1.4$

失败!!!

(d) Unusual: $\begin{cases} f(n) = 3f(\frac{n}{2}) + \Theta(\frac{n}{4}) \\ f(2) = 1 \end{cases} \Rightarrow \Theta(n^{\log_2 3} \lg n)$

Cruel: $f(n) = 2f(\frac{n}{2}) + n^{\log_2 3} \lg n. \Rightarrow \Theta(n^{\log_2 3} \lg n)$

4. (a) 归纳: $n=2$ 时显然成立

假设 $n \le k$ 时成立

$n=k+1$ 时 TPQ$(A[1\ldots k+1], 1, k+1)$ 在运行到 TPQ$(A, 1, q-1)$ 时由归纳假设前 q 项已排好, 此时 $p = q+1$

而后进行第 $q+1$ 项到 $k+1$ 项的排序

由于 $k-q < n$. 由归纳假设, 可以成功排序

(b) $A = [1, 2, 3 \ldots n]$ 即可

(c) 只需保证每次 partition 都分成 n 的常数倍的两部分, 就可以保证 $\Theta(\lg n)$ 的深度 (课上结论引用)

TRQuickSort 代码不变, 修改 Partition 代码即可:

```
NewPartition (A, p, r):
    if (p < r)
        q = Random (p, r)
        swap(A[q], A[r])
    x = A[r], i = p-1
    for (j = p to r-1)
        if (A[j] <= x)
            i = i+1
            swap (A[i], A[j])
    Swap (A[i+1], A[r])
    return i+1
```

5. (a)

```
Sort (A, n)
    for (i = 0 to 2√n-2)        // 每次 i=i0 调用完，前 (i0+2)/2 √n 个已排好
        for (j = i downto 0)
            SqrtSort (j)
```

Correctness : ①

② 每次 $j = j_0$ 调用完，在前 $\frac{j+2}{2}\sqrt{n}$ 项中，

$A[\frac{j}{2}\sqrt{n} \dots \frac{j+2}{2}\sqrt{n}]$ 已排好序

优点：无论 n 是否是 2 的幂，可以保证 SqrtSort 每次恰好排列 $\sqrt{n}$ 个数，而分治法无此优势

调用次数：$1+2+3+\dots+(2\sqrt{n}-1) = \frac{(2\sqrt{n}-1)\cdot 2\sqrt{n}}{2} = 2n-\sqrt{n}$

(b) SqrtSort(k)

```
n = A.size
q = sqrt(n)
Sort(A[k+1 ... k+q])
```

time: $O(n)$

6. (a) 设概率为 p

$$p = \frac{1}{2}(1-p) \Rightarrow p = \frac{1}{3}$$

(b) 数学期望 $E = \frac{1}{2}\times 1 + \frac{1}{4}\times 2 + \frac{1}{8}\times 3 + \cdots + \frac{n}{2^n}$

$$\Rightarrow E = 1 + \frac{1}{2}\times 2 + \cdots + \frac{n}{2^{n-1}}$$

$$E = 2E - E = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 2$$

(c) OneInTwo()

```
if(BiasedCoin()==0)
    if(BiasedCoin()==1)
        return 0
else if(BiasedCoin()==1)
    if(BiasedCoin()==0)
        return 1
else
    return OneInTwo()
```

(d) $E = 2 + \frac{1}{2}E \Rightarrow E = 4$

7.

question: 把那个数转换成20位2进制数，高位不足补0之后，
第i位是0吗？
(i=1~20，一共问20次)
upper bound 与 lower bound 都是20.

Bonus Problem

HeapSort (data[1..n])
heap = BuildMaxHeap (data[1..n]) --- lgn.
for i = n downto 2 --- n
    curmax = heap.HeapExtractMax() --- nlgn
    data[i] = curmax --- n.

⇒ θ(nlgn)