## Problem 1: Hash Collisions - Direct Addressing

Consider a hash table consisting of $m = 11$ slots, and suppose we want to insert integer keys $A = [43, 23, 1, 0, 15, 31, 4, 7, 11, 3]$ orderly into the table.

First, let's suppose the hash function $h_1$ is:

$$h_1(k) = (11k + 4) \mod 10$$

Collisions should be resolved via chaining. If there exists collision when inserting a key, the inserted key(collision) is stored at the end of a chain.

(1) In the table below is the hash table implemented by array. Draw a picture of the hash table after all keys have been inserted. If there exists a chain, please use → to represent a link between two keys.

$h_1(43) = (11 \times 43 + 4) \bmod 10 = 7$ $h_1(23) = (11 \times 23 + 4) \bmod 10 = 7$

$h_1(1) = (11 \times 1 + 4) \bmod 10 = 5$ $h_1(0) = (11 \times 0 + 4) \bmod 10 = 4$

$h_1(15) = (11 \times 15 + 4) \bmod 10 = 9$ $h_1(31) = (11 \times 31 + 4) \bmod 10 = 5$

$h_1(4) = (11 \times 4 + 4) \bmod 10 = 8$ $h_1(7) = (11 \times 7 + 4) \bmod 10 = 1$

$h_1(11) = (11 \times 11 + 4) \bmod 10 = 5$ $h_1(3) = (11 \times 3 + 4) \bmod 10 = 7$

| Index | Keys |
|---|---|
| 0 | |
| 1 | 7 |
| 2 | |
| 3 | |
| 4 | 0 |
| 5 | 1 → 31 → 11 |
| 6 | |
| 7 | 43 → 23 → 3 |
| 8 | 4 |
| 9 | 15 |
| 10 | |

(2) What is the load factor $\lambda$?

$$\lambda = \frac{n}{m} = \frac{10}{11} \approx 0.909$$

(3) Suppose the hash function is modified into $A = [43, 23, 1, 0, 15, 31, 4, 7, 11, 3]$

$$h_2(k) = ((11k + 4) \mod c) \mod 10$$ $B = \{11k+4 \mid k \in A\}$

$B = [477, 257, 15, 4, 169, 345, 48, 81, 125, 37]$

for some positive integer $c$. Find the smallest value of $c$ such that no collisions occur when inserting the keys from $A$. Show you steps in detail and draw a picture of the hash table after all keys have been inserted in the table below. If there exists a chain, please use → to represent a link between two keys.

| Index | Keys |
|---|---|
| 0 | 23 |
| 1 | 15 |
| 2 | 7 |
| 3 | 43 |
| 4 | 0 |
| 5 | 1 |
| 6 | 11 |
| 7 | 3 |
| 8 | 4 |
| 9 | 31 |
| 10 | |

**Steps of finding $c$:** c=79

1°. $0 < c \leq 10$, c is an integer: $(11k+4) \mod c$ can have c possible values $(0, \cdots, c-1)$. According to Pigeonhole Principle, when we have 11 slots, there exist at least (11-c) one slot containing more than one keys. Hence, whe $0 < c \leq 10$, it is impossible to avoid collisions.

2°. $c \geq 11$, c is an integer. Since the load factor $\lambda = 0.909 < 1$, it is possible to have no chain. It is easy to know that c is a prime, then we can try from the smallest prime $\geq 11$.

| $h_2(k)$ B | × ① c=11 | ② c=13 | ③ c=17 | ④ c=19 | ⑤ c=23 | ⑥ c=29 | ⑦ c=31 | ⑧ c=37 | ⑨ c=41 | ⑩ c=43 | ⑪ c=47 | ⑫ c=53 | ⑬ c=59 | ⑭ c=61 | ⑮ c=67 | ⑯ c=71 | ⑰ c=73 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 477 | ④ collide | 9 | 1 | 2 | 7 | 3 | 2 | 3 | 6 | ④ | 7 | 0 | ⑤ | ⓪ | 8 | ① | 9 |
| 257 | ④ X | ⓪ | 0 | ⓪ | ④ | ⑤ | 9 | ⑤ | 1 | | ⑧ | ⑤ | 1 | 3 | 6 | 4 | 8 |
| 15 | | 2 | ⑤ | 5 | 5 | ⑤ | 5 | ⑤ | ⑤ | | 5 | ⑤ | ⑤ | 5 | ⑤ | 5 | 5 |
| 4 | | 4 | 4 | 4 | ④ | | ④ | | 4 | ④ | 4 | 4 | | 4 | 4 | 4 | 4 |
| 169 | | ⓪ | 6 | 7 | | | ④ | | ⑤ | | ⑧ | | | 7 | ⑤ | 7 | ③ |
| 345 | | 7 | ⑤ | 3 | | | | | | | | | | ⓪ | | ① | ③ |
| 48 | | | | ⓪ | | | | | | | | | | | | | |
| | | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

⑱ c=79

| A | B | |
|---|---|---|
| 43 | 477 | 3 |
| 23 | 257 | 0 |
| 1 | 15 | 5 |
| 0 | 4 | 4 |
| 15 | 169 | 1 |
| 31 | 345 | 9 |
| 4 | 48 | 8 |
| 7 | 81 | 2 |
| 11 | 125 | 6 ✓ |
| 3 | 37 | 7 |

Hence, the smallest value of c is 79.

From the previous questions, we can easily find that there often exists collisions in hash table. In real world, collisions is everywhere. What we need to do is to reduce the case of collisions.

Suppose we want to store $n$ items into a hash table with size $m$. Collisions resolved by chaining.

(4) In lecture, Prof. Zhao has mentioned that we often want a hash table which can support `Search()` in $O(1)$ time complexity. Assume simple uniform hashing, that is $E[h(k) = l] = \frac{n}{m}$. Therefore, is `Search()` always has expected running time $O(1)$ as the lecture says? If it always, please explain your reason. If it is not always, under what case can `Search()` for a hash table has expected running time $O(n)$?

Search() doesn't always have expected running time O(1).

When $n >> m$ ( $\lim_{n>>m} \frac{n}{m} = n$ ) , the running time can reach O(n)

After lecture, TA Yuan and TA Xu are all motivated by Prof.Zhao. For this time both of them have two interesting ideas on dealing with collisions to reduce the worst time complexity for each operation.

(5) TA Xu wants to deal with collision via **resizing the array**. He lets $m \to 2m$, but he is so lazy that he doesn't want to change the hash function. Is this method can reduce collisions? Why or Why not?

This method cannot reduce collisions. Since he doesn't change the hash function, the index of the keys remain the same and the situation in (4) still exists.

(6) TA Yuan wants to store each chain using a data structure $S$ instead of a linked list because he think this method can reduce the worst time complexity for insertion. Assume $\Theta(m) = \Theta(n)$, so the load factor $\lambda = \frac{\Theta(n)}{\Theta(m)} = 1$. Remember that in the hash table, there is no duplications. Suppose the data structure $S$ can be **Binary Search Tree**, **Binary Heap** or **AVL Tree**. What is the worst-case running time of `Insert()` for each data structure in new hash table? Explain your reason briefly.

In the worst case:

1° Binary Search Tree : O(n²)

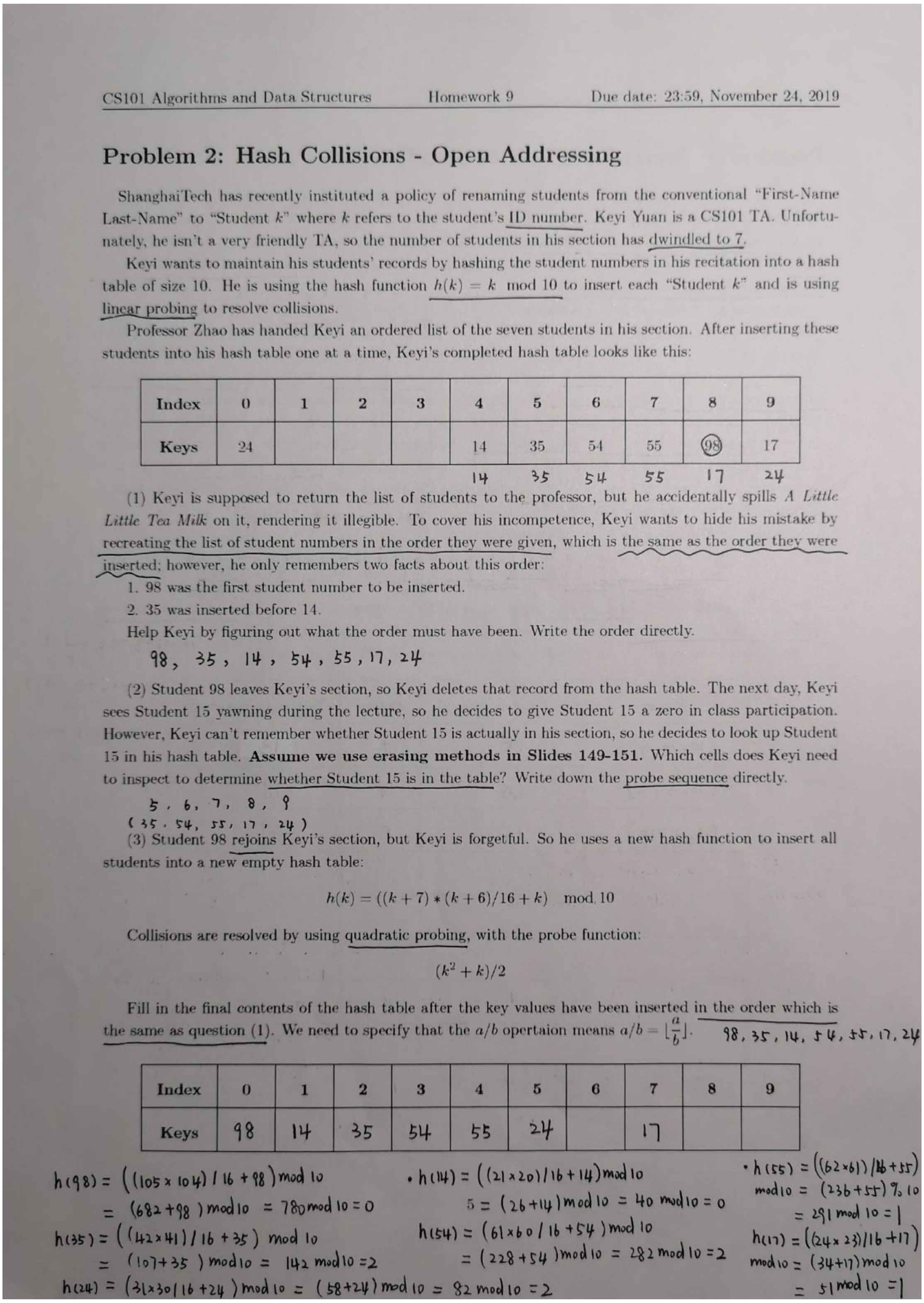
The worst-case running time of inserting a node into the Binary Search Tree is O(n). There are n keys, and if they are all under worst case, then the worst-case running time of Insert() is O(n²).

2° Binary Heap : O(nlogn)

It costs O(logn) to insert a node into the Binary Heap. There are n keys. Hence the worst-case is O(nlogn).

3° AVL tree : O(nlogn)

It costs O(logn) to insert a node into the AVL tree. There are n keys. Hence, the worst-case Insert() is O(nlogn).



## Problem 2: Hash Collisions - Open Addressing

ShanghaiTech has recently instituted a policy of renaming students from the conventional "First-Name Last-Name" to "Student $k$" where $k$ refers to the student's ID number. Keyi Yuan is a CS101 TA. Unfortunately, he isn't a very friendly TA, so the number of students in his section has dwindled to 7.

Keyi wants to maintain his students' records by hashing the student numbers in his recitation into a hash table of size 10. He is using the hash function $h(k) = k \mod 10$ to insert each "Student $k$" and is using linear probing to resolve collisions.

Professor Zhao has handed Keyi an ordered list of the seven students in his section. After inserting these students into his hash table one at a time, Keyi's completed hash table looks like this:

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Keys | 24 | | | | 14 | 35 | 54 | 55 | 98 | 17 |
| | | | | | 14 | 35 | 54 | 55 | 17 | 24 |

(1) Keyi is supposed to return the list of students to the professor, but he accidentally spills *A Little Little Tea Milk* on it, rendering it illegible. To cover his incompetence, Keyi wants to hide his mistake by recreating the list of student numbers in the order they were given, which is the same as the order they were inserted; however, he only remembers two facts about this order:

1. 98 was the first student number to be inserted.
2. 35 was inserted before 14.

Help Keyi by figuring out what the order must have been. Write the order directly.

98, 35, 14, 54, 55, 17, 24

(2) Student 98 leaves Keyi's section, so Keyi deletes that record from the hash table. The next day, Keyi sees Student 15 yawning during the lecture, so he decides to give Student 15 a zero in class participation. However, Keyi can't remember whether Student 15 is actually in his section, so he decides to look up Student 15 in his hash table. **Assume we use erasing methods in Slides 149-151.** Which cells does Keyi need to inspect to determine whether Student 15 is in the table? Write down the probe sequence directly.

5, 6, 7, 8, 9
(35, 54, 55, 17, 24)

(3) Student 98 rejoins Keyi's section, but Keyi is forgetful. So he uses a new hash function to insert all students into a new empty hash table:

$$h(k) = ((k + 7) * (k + 6)/16 + k) \mod 10$$

Collisions are resolved by using quadratic probing, with the probe function:

$$(k^2 + k)/2$$

Fill in the final contents of the hash table after the key values have been inserted in the order which is the same as question (1). We need to specify that the $a/b$ opertaion means $a/b = \lfloor \frac{a}{b} \rfloor$. 98, 35, 14, 54, 55, 17, 24

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Keys | 98 | 14 | 35 | 54 | 55 | 24 | | 17 | | |

$h(98) = ((105 \times 104) / 16 + 98) \bmod 10$
$= (682 + 98) \bmod 10 = 780 \bmod 10 = 0$

$h(35) = ((42 \times 41) / 16 + 35) \bmod 10$
$= (107 + 35) \bmod 10 = 142 \bmod 10 = 2$

$h(24) = (31 \times 30 / 16 + 24) \bmod 10 = (58 + 24) \bmod 10 = 82 \bmod 10 = 2$

$\cdot\ h(14) = ((21 \times 20) / 16 + 14) \bmod 10$
$5 = (26 + 14) \bmod 10 = 40 \bmod 10 = 0$

$h(54) = (61 \times 60 / 16 + 54) \bmod 10$
$= (228 + 54) \bmod 10 = 282 \bmod 10 = 2$

$\cdot\ h(55) = ((62 \times 61) / 16 + 55)$
$\bmod 10 = (236 + 55) \% 10$
$= 291 \bmod 10 = 1$

$h(17) = ((24 \times 23) / 16 + 17)$
$\bmod 10 = (34 + 17) \bmod 10$
$= 51 \bmod 10 = 1$

## Problem 3: Secret Love In ShanghaiTech

Bob and Alice are sophomores in ShanghaiTech. Bob loves Alice. But Alice doesn't know he loves her. And Bob is too shy to express his love. Therefore, he wants to send Alice a secret message of characters via a specially encoded sequence of integers. Each character of Bob's message corresponds to a **satisfying** triple of distinct integers from the sequence satisfying the following property: every permutation of the triple occurs consecutively within the sequence.

0 1 2 3 4 5 6 7 8 9 ✓ 10 11 12 13

For example, if Bob sends the sequence $A = (4, 10, 3, 4, 10, 9, 3, 10, 4, 3, 10, 4, 8, 3)$, then the triple $t = \{3, 4, 10\}$ is satisfying because every permutation of $t$ appears consecutively in the sequence, i.e. $(3, 4, 10)$, $(3, 10, 4)$, $(4, 3, 10)$, $(4, 10, 3)$, $(10, 3, 4)$ and $(10, 4, 3)$ all appear as consecutive sub-sequences of $A$.

For any satisfying triple, its **initial occurrence** is the smallest index $i$ such that all permutations of $t$ is shown from index 0 to $i$ in the sequence. For example, the initial occurrence of $t$ in $A$ is 10.

Bob's secret message will be formed by characters corresponding to each satisfying triple in the encoded sequence, increasingly ordered by initial occurrence. To convert a satisfying triple $(a, b, c)$ into a character, Bob sends Alice an arithmatic function $f(a, b, c) = (a+b+c) \mod 27$ to decode the message. $f(\cdot) = 0$ to 25 correspond to the lower-case letters 'a' to 'z', while $f(\cdot) = 26$ corresponds to a space character. For example, the triple $\{3, 4, 10\}$ corresponds to the letter 'r'.

(3+4+10) mod 27 = 17 r

(1) Suppose Bob sends Alice the following sequence:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

$$(10, 13, 9, 10, 13, 5, 2, 13, 5, 10, 9, 13, 10, 9, 13, 2, 5, 13, 2, 67, 23, 1, 2, 10, 1, 2, 1, 10, 2, 1)$$

Write down the list of satisfying triples contained in the sequence, as well as their associated characters, ordered by initial occurrence. What is the secret message? Fill in the table below.

You may not fill in all rows of the table, or you may need to add rows of the table.

10 + 13 + 9 = 32 mod 27
= 5

13+5+2 = 20 mod 27
= 20

1+2+10 = 13 mod 27
= 13

| Triple | $(a+b+c) \mod 27$ | Letter | Initial Occurrence |
|---|---|---|---|
| (10, 13, 9) | 5 | f | 13 |
| (13, 5, 2) | 20 | u | 18 |
| (1, 2, 10) | 13 | n | 29 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

(2) Now Bob has the courage to express his love to Alice. But Alice refuses him immediately, because she is tired of decoding his messages by hand. After that, Bob becomes very sad, knowing that Alice is very angry about the process finding all satisfying triples, which needs $O(n^3)$ time complexity. In order to help him cheer up, please design an expected $O(n)$ time algorithm to decode a sequence of $n$ integers from Bob to output his secret message. Hint: Hash Table.

(If we have n numbers, then the number of triples is n-2 : m = n-2)

$f(a,b,c) = (a+b+c) \bmod 27$

Suppose we have m triples, considering a hash table consisting of m slots, and the hash function is:

$$f(a,b,c) = (a+b+c) \bmod 27$$

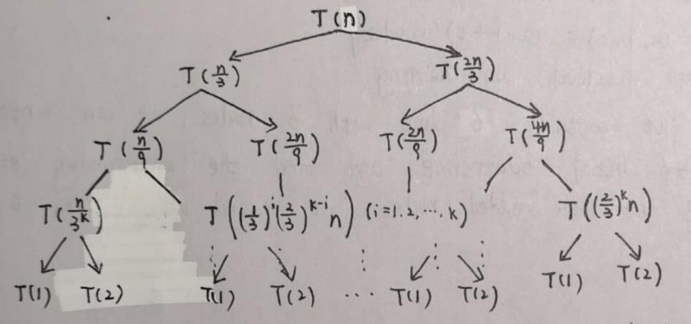
Collisions are resolved via chaining.

When a slot contains a chain with 6 nodes, we can write down the initial occurrence and find the corresponding letter. (Ignoring the later added node, just record when it has 6 nodes)

## Problem 4: Inequal Divide

In this problem, we will analyze an alternative to divide step of merge sort. Consider an algorithm `InequalSort()`, identical to `MergeSort()` except that, instead of dividing an array of size $n$ into two arrays of size $\frac{n}{2}$ to recursively sort, we divide into two arrays with inequal size. For simplicity, assume that n is always divisible by divisors (i.e. you may ignore floors and ceilings).

(1) **Analysis:** If we divide into two arrays with sizes roughly $\frac{n}{3}$ and $\frac{2n}{3}$. Write down a recurrence relation for `InequalSort()`. Assume that merging takes $\Theta(n)$ time. Show that the solution to your recurrence relation is $O(n \log n)$ by drawing out a recursion tree, assuming $T(1) = O(1)$. Note, you need to prove both upper and lower bounds.



$n$ $k=0$

$\frac{n}{3} + \frac{2}{3}n$ $k=1$

$\frac{n}{9} + 2 \times \frac{2n}{9} + \frac{4n}{9}$ $k=2$

…

$\sum_{i=0}^{k} \binom{k}{i} \frac{2^i n}{3^k}$ $k$

…

$\frac{n}{3^k} = 1$

$k = \log_{\frac{1}{3}} \frac{1}{n}$

$= \log_3 n$

Note: The T(1)s. T(2)s above may not in the same level.

while $h = \Theta(\log n)$

$\log_3 n \le k \le \log_{\frac{3}{2}} \frac{n}{2}$

$(\frac{2}{3})^k n = 2$

$k = \log_{\frac{2}{3}} \frac{2}{n}$

$h = \log_{\frac{2}{3}} \frac{2}{n} = \Theta(\log n)$

$$T(n) = \begin{cases} O(1) & , \text{if } n=1 \\ T(\frac{n}{3}) + T(\frac{2n}{3}) + \Theta(n) & . \text{otherwise} \end{cases}$$

Hence InequalSort costs $O(n \log n)$

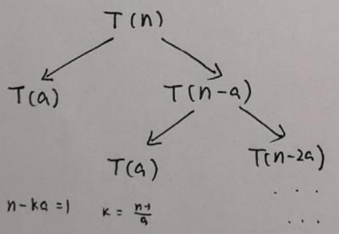
(2) **Generalization:** If we divide array into two arrays of size $\frac{n}{a}$ and $\frac{(a-1)n}{a}$ for arbitrary constant $1 < a$ recursively, what is the asymptotic runtime of the algorithm? Is there any change on time complexity?

$\log_a n \leq k \leq \log_{\frac{a}{a-1}} \frac{n}{2}$

$a \to \infty$ . $\log_{\frac{a}{a-1}} \frac{n}{2} \to n$

The time complexity is $O(n\log n)$ (not change)

(3) **Limitation:** If we divide array into two arrays of size $a$ and $n-a$ for some positive constant integer $a$ recursively, what is the asymptotic runtime of the algorithm? Is there any change on time complexity? Assume that merging still takes $\Theta(n)$ time, $T(a) = O(a)$. It may help to draw a new recursion tree.



$n - ka = 1$ $k = \frac{n-1}{a}$

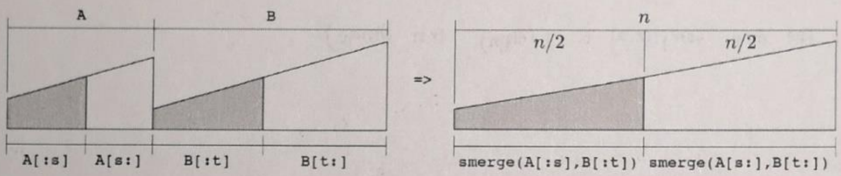
$$T(n) = T(a) + T(n-a) + n$$
$$= \cdots = kT(a) + k\frac{n+a}{2}$$
$$\approx T(ka) + \frac{n^2}{2a} = O(n^2)$$

$h = \lfloor \frac{n}{a} \rfloor \Rightarrow h = O(n)$ Asymptotic runtime is $O(n^2)$.

Time complexity is $O(n^2)$ (change)

## Problem 5: Slice Merge

In this problem, we will analyze an alternative to merge step of merge sort. Suppose $A$ and $B$ are sorted arrays with possibly different lengths, and let $n$ = `len(A)+len(B)`. You may assume $n$ is a power of two and all $n$ items have distinct keys. The slice merge algorithm, `smerge(A,B)`, merges $A$ and $B$ into a single sorted arrays as follows:



**Step 1:** Find indices $s$ and $t$ such that $s+t=\frac{n}{2}$ and prefix subarrays `A[:s]` and `B[:t]` together contain the smallest $\frac{n}{2}$ keys from $A$ and $B$ combined.

**Step 2:** Recursively compute `X = smerge(A[:s], B[:t])` and `Y = smerge(A[s:], B[t:])`, and return their concatenation `X + Y`, a sorted array consisting of all items from $A$ and $B$.

For example, if $A=[1,3,4,6,8]$ and $B=[2,5,7]$, we find $s=3$ and $t=1$ and then recursively compute:

`smerge([1,3,4], [2]) + smerge([6,8], [5,7]) = [1,2,3,4] + [5,6,7,8]`

(1) Describe an algorithm to find indices $s$ and $t$ satisfying step (1) in $O(n)$ time, using only $O(1)$ additional space beyond array $A$ and $B$ themselves. Remember to argue the correctness and running time of your algorithm.

1. Traverse the array ⇒ number of the keys = n

2. find s and t

```
int s=0, p=0;
while (s+t < n/2)
{
    if (A[s] < A[t])
    {
        ++s;
        continue;
    }
    else
    {
        ++t;
        continue;
    }
}
```

Use O(1) additional space to store s and t.

Time complexity: O(n) (Just traverse and compare $\frac{n}{2}$ times)

(2) Write and solve a recurrence for $T(n)$, the running time of `smerge(A,B)` when $A$ and $B$ contain a total of $n$ items. Please show your steps. How does this running time compare to the `merge` step of `MergeSort()`?

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + O(n)$ $\quad \exists c.\overset{s.t.}{} T(n) \leq cn\log n$

$T(n) = 2c\frac{n}{2}\log\frac{n}{2} + n = cn\log n - cn + n = cn\log n - n(c-1)$

$\exists c > 1$. s.t. $T(n) \leq cn\log n$

$\Rightarrow$ Time complexity $= O(n\log n)$

Merge Sort: $O(n)$

smerge (A.B) is slower than MergeSort ( ).

(3) Let `smerge_sort(A)` be a variant of `MergeSort(A)` that uses `smerge` in place of `merge`. Write and solve a recurrence for the running time of `smerge_sort(A)`. Please show your steps.

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + O(n\log n)$

Guess $T(n) = O(n\log^2 n)$

$\exists c$ $T(n) \leq cn\log^2 n$

$T(n) \leq 2c\frac{n}{2}\log^2\frac{n}{2} + n\log n = cn(\log n - 1)^2 + (n\log n)$

$\leq cn\log^2 n + (n\log n) - 2cn\log n + cn$

$< cn\log^2 n$

$\Rightarrow T(n) = O(n\log^2 n)$