Problem Set 1

Name: BUAA-TYZ

Collaborators: ZR!

Problem 1-1.

(a) Known: $f_1 = nlog(n), f_2 = (logn)^n, f_3 = 6606log(n), f_4 = (logn)^{6606}, f_5 = loglog(6606n)$

- $\lim_{n\to\infty} \frac{f_1}{f_2} = \lim_{n\to\infty} e^{ln\frac{n}{(logn)^{n-1}}} = \lim_{n\to\infty} e^{lnn-(n-1)*lnlogn}$
- Obviously, $\lim_{n\to\infty} lnlogn > 1$ *and* $lnn < (n-1)$.
- We get $\lim_{n\to\infty} \frac{f_1}{f_2} = 0$, which means that $f_1 = o(f_2)$
- Because the monotone of log, we get the answer: $\{f_5, f_3, f_4, f_1, f_2\}$

(b) Known: $f_1 = 2^n, f_2 = 6006^n, f_3 = 2^{6606^n}, f_4 = 6606^{2^n}, f_5 = 6606^{n^2}$

- Compare $f_2, f_4, f_5 \Leftrightarrow n, 2^n, n^2$, we get $\{f_2, f_5, f_4\}$ at first.
- $\lim_{n\to\infty} \frac{f4}{f3} = \lim_{n\to\infty} e^{lnf_4-lnf_3} = \lim_{n\to\infty} e^{2^n ln6606-6606^n ln2} = 0$.
- Finally, we get the answer: $\{f_1, f_2, f_5, f_4, f_3\}$.

(c) Known: $f_1 = n^n, f_2 = \binom{n}{n-6} = O(n^6), f_3 = (6n)!, f_4 = \binom{n}{n/6} = O(n^n), f_5 = n^6$, which we already get $\{f_2, f_5\}, \{f_1, f_4\}$

- $\frac{f_1}{f_3} = \frac{n^n}{6n*6(n-1)...(5n+1)(5n)!} < \frac{1}{(5n)!} \xrightarrow[n\to\infty]{} 0$.
- Finally, we get the answer: $\cancel{\{\{f_2, f_5\}, \{f_1, f_4\}, f_3\}}$

Solution: Stirling's approximation: $n! \simeq \sqrt{2\pi n}(\frac{n}{e})^n$.

$$f_4 = \binom{n}{n/6} = \frac{n!}{(\frac{n}{6})!(\frac{5n}{6})!} \tag{1}$$

$$\simeq \frac{\sqrt{2\pi n}n^n}{2\pi n(\frac{n}{6})^n(5)^{\frac{5n}{6}}} \quad by\ using\ Stirling'sapproximation \tag{2}$$

$$\simeq \frac{6^n}{\sqrt{n}(5)^{\frac{5n}{6}}} \tag{3}$$

We can conclude that $f_4 < f_1$

(d) Known $f_1 = n^{n+4} + n!, f_2 = n^{7\sqrt{n}}, f_3 = 4^{3nlogn}, f_4 = 7^{n^2}, f_5 = n^12 + 1/n$

- We get f_5, f_2 and f_3, f_4 at first. Use the same method as above, we get f_2, f_3
- $n! = e^{ln1 + 1n2 + ... + 1nn} < e^{nlnn}$ and $n^{n+4} = e^{(n+4)lnn}$
- As consequence, $f_1 = O(e^{nlnn}) < f_3$. We get the answer: $\{f_5, f_2, f_1, f_3, f_4\}$

Problem 1-2.

(a)
- Algorithm description: The naive idea is that we delete the k elements and store them in a buffer. Then insert them reversely back into D.
- Time analysis: $2klogn = O(klogn)$
- Optimization: A better idea for this is to swap elements at index i and index i + k - 1, which only needs one loop.

```
def reverse(D, i, k):
    if k <= 1:
        return
    back = D.delete_at(i + k - 1)
    front = D.delete_at(i);
    D.insert_at(i, back)
    D.insert_at(i + k - 1, front)
    reverse(D, i + 1, k - 2)
```

(b)
- Algorithm description: If $i < j$, then we are already done. Otherwise, we move directly by copying elements one by one.
- Time analysis: $2klogn = O(klogn)$

```
def move(D, i, k, j):
    if k <= 0 || i < j :
        return
    for _ in range(0, k):
        x = D.delete_at(i + k - 1)
        D.insert_at(j, x)
```

Problem 1-3.

- Algorithm description: Use a dynamic array before A. Use two deques between A and B and after B.
- Time analysis:
  - build(X): $O(|X|)$. just insert one by one.
  - place mark(i, m): worst case: $O(n)$. Put the bookmark will force the array or deque to move all data before or after index i to another array or deque.
  - read page(i): $O(1)$. Obviously.
  - shift mark(m, d): $O(1)$. Shift mark with one step means move a element in deque to another array or deque. Popback of array and popfront of deque cost constant time. Pushfront or pushback of deque also cost constant time.

–move page(m): $O(1)$. Same reason as above.

Problem 1-4.

(a)
- insert_first(x): If L.head is None, which means L is empty. Then let L.head and L.tail point to x. Otherwise, let x.next point to L.head and L.head.prev point to x. Then modify L.head to x.
- inser_last(x): If L.tail is None, which means L is empty. Then let L.head and L.tail point to x. Otherwise, let x.prev point to L.tail and L.tail.next point to x. Then modify L.tail to x.
- delete_first(): Let L.head point to L.head.next. Modify L.head.prev to None. (For python, there is no need to delete L.head mannully beacause of GC)
- delete_last(): Let L.tail point to L.tail.prev. Modify L.tail.next to None.

(b)
1. Create a double-linked list L', whose head and tail are x1 and x2 separately.
2. Let x1.prev.next point to x2.next
3. If x2.next is not None, then let x2.next.prev point to x1.prev.
4. Set L'.head.prev and L'.tail.next to None.

(c)
1. Let L2.head.prev point to x and L2.tail.next point to x.next
2. Let x.next point to L2.head
3. If L2.tail.next is not None, then let L2.tail.next.prev point to L2.tail
4. Set L2.head and L2.tail to None.

(d) Submit your implementation to `alg.mit.edu`.