Question: https://leetcode.com/problems/create-maximum-number/

To find the maximum ,we can enumerate how digits we should get from nums1 , we suppose it is i.

So , the digits from nums2 is K - i.

And we can use a stack to get the get maximum number(x digits) from one array.

OK, Once we choose two maximum subarray , we should combine it to the answer.

It is just like merger sort, but we should pay attention to the case: the two digital are equal.

we should find the digits behind it to judge which digital we should choose now.

In other words,we should judge which subarry is bigger than the other.

That's all.

Code:

public class Solution {

public int[] maxNumber(int[] nums1, int[] nums2, int k) {

int get\_from\_nums1 = Math.min(nums1.length, k);

int[] ans = new int[k];

for (int i = Math.max(k - nums2.length, 0); i <= get\_from\_nums1; i++) {

int[] res1 = new int[i];

int[] res2 = new int[k - i];

int[] res = new int[k];

res1 = solve(nums1, i);

res2 = solve(nums2, k - i);

int pos1 = 0, pos2 = 0, tpos = 0;

while (res1.length > 0 && res2.length > 0 && pos1 < res1.length && pos2 < res2.length) {

if (compare(res1, pos1, res2, pos2))

res[tpos++] = res1[pos1++];

else

res[tpos++] = res2[pos2++];

}

while (pos1 < res1.length)

res[tpos++] = res1[pos1++];

while (pos2 < res2.length)

res[tpos++] = res2[pos2++];

if (!compare(ans, 0, res, 0))

ans = res;

}

return ans;

}

public boolean compare(int[] nums1, int start1, int[] nums2, int start2) {

for (; start1 < nums1.length && start2 < nums2.length; start1++, start2++) {

if (nums1[start1] > nums2[start2]) return true;

if (nums1[start1] < nums2[start2]) return false;

}

return start1 != nums1.length;

}

public int[] solve(int[] nums, int k) {

int[] res = new int[k];

int len = 0;

for (int i = 0; i < nums.length; i++) {

while (len > 0 && len + nums.length - i > k && res[len - 1] < nums[i]) {

len--;

}

if (len < k)

res[len++] = nums[i];

}

return res;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>