Homework9

软件13 杨楠

1

a.

证明：设一个最优解序列。对于其中任意一个任务 a_j ，记这个任务所在的时间槽的位置为 p_j 。
如果 $p_j \le d_j$ ，那么总可以将 a_j 与任何一个所在时间槽位置大于 p_j ，小于等于 d_j 的任务进行位置交换，而且总惩罚不会增加。

如果 $p_j > d_j$ ，那么 a_j 之后的任何任务，都满足截止时间大于 d_j ，否则，可将这个任务与 a_j 交换，总惩罚减少，这与原序列是最优解矛盾。所以，可将 a_j 通过交换，移动到尽量后面，且总惩罚不会增加。

综上，该算法总能得到最优解。

b.

构建不相交集合森林，每个集合存储的是安排好了的时间相连的任务。对于集合的代表 x ，用 $early[x]$ 和 $late[x]$ 表示该集合的任务的最早和最晚时间。每次遍历任务时，按照a中的算法填入时间槽后，如果该时间槽两边非空，则可以进行UNION。

记任务 a_i 在时间槽中的位置为 p_i 。指针数组 $ptr[i]$ 指向截止时间为i的任务。使用不相交集合森林的算法伪代码如下。

```
SCHEDULE(A)
    let ptr[1..n] be a new array
    for i = 1 to n
        p[i] = d[i]
        if ptr[d[i]] != NIL
            m = FIND-SET(ptr[d[i]])
            p[i] = early[m] - 1
        x = MAKE-SET(i)
        ptr[p[i]] = x
        early[x] = p[i]
        late[x] = p[i]
        if ptr[p[i] - 1] != NIL
            UNION(ptr[p[i]], ptr[p[i] - 1])
        if ptr[p[i] + 1] != NIL
            UNION(ptr[p[i]], ptr[p[i] + 1])
```

每次遍历时，调用1次MAKE-SET，最多调用2次UNION，由21.3节中的不相交集合森林的算法可得，上述程序所需运行时间为 $\mathrm{O}(n\alpha(n))$ 。

2

a.

设计SEARCH算法如下：对于这 k 个数组，依次遍历，如果非空，则对该数组进行一次二分查找的操作，直到查找成功。

由于每个 A_i 的元素个数为 2^i ，二分查找该数组所需时间为 $\mathrm{O}(\lg 2^i) = \mathrm{O}(i)$ 。最坏情况下，每个数组都是满的，总的运行时间为 $T(n) = \sum_{i=0}^{k-1} \mathrm{O}(i) = \frac{ck(k-1)}{2} = \mathrm{O}(\lg^2 n)$ 。

b.

设计INSERT算法如下。每次插入一个新元素时，如果 A_0 为空，那么直接将这个新元素填入即可。如果 A_0 为满，那么将 A_0 以及之后所有连续的满数组，连同这个新元素，合并为一整个数组，填补代替前一位空数组，再将原先这些连续的满数组置空。

最坏情况：这 k 个数组都是满的，那么插入一个新元素，要将这 k 个数组都合并。对于两个各自有序的数组进行合并，所需时间与数组长度成线性关系，所以总时间为

$$\sum_{i=0}^{k-1} \mathrm{O}(2^i) = \mathrm{O}(2^k) = \mathrm{O}(n)$$

均摊时间：每次使用INSERT算法的过程，相当于相应的 k 位二进制数加1。数组的从满变空，以及从空变满的过程，对应了该位数从1变0，从0变1的翻转过程。

从空集合加入 n 个元素，即从头执行 n 次INSERT算法，对应了二进制数从0逐次加1至n的过程。由算法操作可知，某位数字的每次翻转（此时进行的是数组合并的过程），所需时间都是 $\mathrm{O}(2^i)$ （包括最低位的从0置1，所需时间是 $\mathrm{O}(1)$ ）。整个过程中，最低位数组 A_0 翻转了 n 次，A_1 翻转了 $\lfloor \frac{n}{2} \rfloor$ 次，……每位数组 A_i 翻转了 $\lfloor \frac{n}{2^i} \rfloor$ 次，从而全过程的时间开销为

$$\sum_{i=0}^{k-1} \lfloor \frac{n}{2^i} \rfloor \mathrm{O}(2^i) = \mathrm{O}(nk) = \mathrm{O}(n \lg n)$$

从而均摊到每一次所用的时间为 $\mathrm{O}(\lg n)$ 。

c.

记当前位数最小的满数组为 A_m ，显然，所要删除的元素所在的位置数组，下标一定大于等于 m ，如果该元素就在 A_m ，那么删除这个元素；否则，删除这个元素后，从 A_m 中取一元素填补到那个数组的正确的大小顺序。无论如何， A_m 中都会剩余 $2^m - 1$ 个元素。将这些元素按顺序划分为 m 组，每组元素的个数分别为 $2^0, 2^1, \ldots, 2^{m-1}$ ，分别作为数组 $A_0, A_1, \ldots, A_{m-1}$ 中（即分别填满），而 A_m 数组就变为空。由于 A_m 本身有序，上述划分之后，每组的内部显然有序。如果DELETE函数传入的参数是某个具体的数值，那么首先是SEARCH这个数，所需时间为 $\mathrm{O}(\lg^2 n)$ 。删除该数所需时间（最坏）为 $\mathrm{O}(\lg k)$ ，拆分和填满的所需时间为 $\mathrm{O}(2^{k-1})$ （对应最坏情况：最高位是1，后面位数都是0），总的时间为

$$\mathrm{O}(\lg^2 n) + \mathrm{O}(\lg k) + \mathrm{O}(2^{k-1}) = \mathrm{O}(n)$$