Πολύχρωμη κορδέλα

Μία προφανής λύση της άσκησης είναι να εξετάσουμε για κάθε δυνατό κομμάτι της κορδέλας, έστω το διάστημα $[i, j)$, αν αυτό περιέχει όλα τα χρώματα. Τα δυνατά κομμάτια είναι $O(N^2)$ και ο έλεγχος κοστίζει $O(N)$, άρα η συνολική πολυπλοκότητα αυτής της λύσης είναι $O(N^3)$. Μία εύκολη βελτιστοποίηση είναι να εξετάζουμε συγχρόνως όλα τα κομμάτια που ξεκινούν από το ίδιο σημείο i και όπως αυξάνεται το j να σημειώνουμε τα χρώματα που έχουμε δει. Με τον τρόπο αυτό, η συνολική πολυπλοκότητα της λύσης γίνεται $O(N^2)$. Και οι δύο αυτές λύσεις δεν είναι αρκετές για να πάρουν το 100% των μονάδων αυτής της ασκησης.

Για μία καλύτερη λύση, μπορούμε να χρησιμοποιοιήσουμε μία "άπληστη" (greedy) προσέγγιση. Εξετάζουμε το διάστημα $[i, j)$ και ξεκινάμε με $i = j = 0$. Μετράμε τις εμφανίσεις κάθε χρώματος στο διάστημα αυτό. Διατηρούμε ένα μετρητή για κάθε χρώμα, που αρχικά είναι ίσος με 0, και επίσης διατηρούμε το πλήθος των χρωμάτων που εμφανίζονται, πάλι αρχικά ίσο με 0. Όσο στο διάστημα $[i, j)$ δεν εμφανίζονται όλα τα χρώματα, αυξάνουμε το j ενημερώνοντας κατάλληλα τους μετρητές και το πλήθος των εμφανιζόμενων χρωμάτων. Μόλις εμφανιστούν όλα τα χρώματα και όσο συνεχίζουν να εμφανίζονται, αυξάνουμε το i, πάλι ενημερώνοντας κατάλληλα τους μετρητές και το πλήθος των εμφανιζόμενων χρωμάτων. Η ελάχιστη τιμή του $j - i$ τέτοια ώστε να εμφανίζονται όλα τα χρώματα είναι η σωστή απάντηση.

Στην περίπτωση γλωσσών προστακτικού ή αντικειμενοστρεφούς προγραμματισμού με μεταβλητές που οι τιμές τους μεταβάλλονται, μπορούμε να χρησιμοποιήσουμε έναν πίνακα (array) για τους μετρητές. Αυτό απαιτεί κόστος $O(1)$ για την ενημέρωση των μετρητών. Επομένως, η πολυπλοκότητα της λύσης είναι $O(N)$ γιατί στη χειρότερη περίπτωση οι δύο δείκτες i και j θα αυξηθούν από 0 μέχρι N.

Στην περίπτωση γλωσσών αγνού συναρτησιακού προγραμματισμού, για τους μετρητές μπορούμε να χρησιμοποιήσουμε μία κατάλληλη δομή δεδομένων, π.χ. ένα ordered map, που συνήθως υλοποιείται με κάποιο ισοζυγισμένο δένδρο δυαδικής αναζήτησης και έχει κόστος ενημέρωσης $O(KlogK)$.

- Ordered map σε SML/NJ: BinaryMapFn
- Ordered map σε OCaml: Map

Επομένως, σε μία γλώσσα αγνού συναρτησιακού προγραμματισμού, η παραπάνω λύση θα έχει πολυπλοκότητα $O(NlogK)$, αρκετή για να πάρει το 100% των μονάδων αυτής της άσκησης.