دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی کامپیوتر

طراحی الگوریتم ها ـ دکتر احمدی

نیم‌سال دوم ۹۹ـ۰۰

پروژه دوم

محمّد نامورپور ۹۹۲۰۳۵۴

پاراگراف ها

برای حل این سوال از روش برنامه نویسی پویا [۱] استفاده کرده ام.
ابتدا هزینه ی تمام خطوط ممکن را در یک ماتریس به نام lineCost ذخیره می کنیم.برای این کار از فرمول ۱ استفاده می کنیم.

$$lineCost(i,j) = (M - (i-j) - \sum_{k=i}^{j} L_k)^3 \quad (۱)$$

مقدار $lineCost[i][j]$ نشان می دهد که اگر کلمات iام تا jام در یک خط قرار بگیرند، هزینه چقدر خواهد بود. اگر کلمات iام تا jام نتوانند در یک خط قرار بگیرند (یعنی تعداد کاراکترهایشان از ماکزیمم بیشتر باشد) مقدار $lineCost[i][j]$ برابر با بی نهایت در نظر گرفته می شود، [۲] تا جزئی از راه حل نباشد.
وقتی ماتریس lineCost را تکمیل کردیم، هزینه کل را از فرمول بازگشتی زیر باید تعیین کنیم.

$$totalCost(j) = \begin{cases} 0, & j = 0 \\ min(lineCost[i-1] + l_c[i,j]), & j > 0 \end{cases}$$

همانطور که می بینید در فرمول بازگشتی بالا، مثلا پاسخ زیرمسئله $totalCost(2)$ توسط $totalCost(3)$ و $totalCost(4)$ استفاده می شود. برنامه نویسی پویا برای ذخیره سازی پاسخ های زیرمسئله ها به کار گرفته شده است.
برای چاپ کردن خروجی، مشخص میکنیم که کدام کلمات در کدام خطوط قرار میگیرند. برای این منظور از لیست $solution$ استفاده کرده ایم که نشان می دهد هر مقدار در لیست $totalCost$ از کجا آمده است.

پیچیدگی زمانی

در قسمتی که میخواهیم ماتریس $lineCost$ را تکمیل کنیم، دو حلقه تو در تو داریم که هر کدام از 0 تا $n+1$ را می پیمایند.
پس در صورتی که n کلمه به الگوریتم داده شود، پیچیدگی زمانی الگوریتم نیز $O(n^2)$ خواهد بود.

---

[1] Dynamic Programming

[۲] در کد برای راحتی مقدار بی نهایت را برابر ۹۹۹۹۹۹۹۹۹ در نظر گرفتم