Algorytmy kombinatoryczne w bioinformatyce − Zadanie IV

Parametry wejściowe:
akwb-zadanie-4.exe nazwa_pliku
np.
akwb-zadanie-4.exe testy/ins-PDP-11a-asc.txt

Opis algorytmu:

1. **Wczytanie sekwencji z plików wejściowych**
   Otwieramy plik o nazwie podanej w parametrze wejściowym, ładujemy jego zawartość do pamięci.

2. **Sprawdzenie ilości cięć**
   Celem uniknięcia niepotrzebnych obliczeń sprawdzamy czy z liczby wczytanych fragmentów jesteśmy w stanie skonstruować mapę. Przekształcając symbol Newtona jesteśmy w stanie wstępnie zweryfikować poprawność danych wejściowych.

$$|A| = \binom{k+2}{2} = \frac{(k+2)!}{2k!} = \frac{(k+1)(k+2)}{2} = \frac{k^2+3k+2}{2}$$

$$k = \frac{-b+\sqrt{\Delta}}{2a} = \frac{-3+\sqrt{9-4\cdot(2-2\cdot|A|)}}{2} = \frac{-3+\sqrt{1+8\cdot|A|}}{2}$$

   Według powyższego, na podstawie ilości fragmentów weryfikujemy czy istnieje liczba całkowita *k* z której można można wygenerować multizbiór *A* o takiej samej liczności jak ten podany na wejściu. W przeciwnym wypadku oznacza to błąd w danych, wypisujemy wtedy odpowiedni komunikat i nie przystępujemy do dalszych obliczeń.

3. **Wyszukiwanie mapy**
   Tworzymy listę przechowującą elementy mapy, na pierwszę pozycję wstawiamy różnicę dwóch największych fragmentów. Wywołujemy funkcję *find* która przyjmuje w parametrach odległość do prawego końca, listę nieużytych jeszcze fragmentów oraz mapę. Odległość do prawego końca ustawiana jest na najdłuższy fragment, z każdym dodaniem elementu to mapy jest ona zmniejszana o jego długość.

   A) Na początku funkcji sprawdzamy czy w zbiorze nieużytych jeszcze fragmentów znajduje się element o długości podanej w parametrze, tzn. o odległości do prawego końca. Jeżeli został znaleziony oznaczamy go jako odwiedzony i przechodzimy dalej, jeśli takiego elementu nie ma wychodzimy z funkcji.

   B) Sprawdzamy czy po powyższym usunięciu zostały jeszcze jakieś nieodwiedzone elementy. Jeśli tak, to przechodzimy dalej, w innym wypadku oznacza to że mamy już gotowe rozwiązanie, zwracamy gotową mapę.

C) Obliczamy sumy wszystkich fragmentów od prawej strony obecnej mapy, wyszukujemy je i oznaczamy je jako odwiedzone.

Na przykład:

```
mapa = (3,  4,  2)
                2 = oznaczamy 2 jako odwiedzone
            4 + 2 = oznaczamy 6 jako odwiedzone
        3 + 4 + 2 = oznaczamy 9 jako odwiedzone
```

Jeżeli któregoś z elementów nie ma w liście nieodwiedzonych elementów wychodzimy z funkcji.

D) Od obecnej odległości do prawego końca odejmujemy długość ostatniego elementu w mapie.

E) Dla każdego nieodwiedzonego jeszcze elementu wywołujemy rekurencyjnie funkcję *find*. Do parametrów przekazujemy odległość do prawego końca z punktu D, kopię z listą nieodwiedzonych elementów i mapę do której dodajemy przetwarzany właśnie element.

Po wykonaniu sprawdzamy czy wywoływana rekurencyjnie funkcja zwróciła prawidłową mapę, jeżeli tak wychodzimy z obecnego wywołania zwracając ten wynik, w przeciwnym wypadku kontynuujemy iterowanie po nieodwiedzonych elementach, gdy elementy zostały wyczerpane wychodzimy z funkcji.

4. **Wypisanie wyników wyszukiwania**

Wnioski:
Czas wykonywania algorytmu zależy w głównej mierze od długości fragmentów w danych wejściowych. Im więcej krótkich, powtarzających się fragmentów fragmentów, tym dłużej trwa wykonywanie programu.

Np. test **ins-PDP-14a-asc.txt** ze 120 elementami fragmentu DNA o długości 81 wykonuje się 22 minuty, test **ins-PDP-14b-asc.txt** z taką samą ilością elementów ale z większym “rozstawem”, (całkowita długość 673) wykonuje się 0.01 sekundy.

Uporządkowanie danych wejściowych nie ma znaczącego wpływu na algorytm gdyż są one na początku sortowane.

Testy:
> test_01.txt

```
3 6 9 9 12 15 15 18 18 21 21 24 27 27 27 30 30 33 33 36 36 39 39 42 42 45
45 45 45 51 54 54 57 60 63 63 63 66 69 72 75 75 78 81 81 81 84 87 90 90 90
99 99 102 105 105 108 108 114 117 117 120 126 126 126 132 135 135 135 138
147 150 150 153 156 162 162 165 165 168 171 180 180 189 189 189 195 195
198 204 207 210 216 225 225 228 231 231 234 243 252 252 255 264 270 270
273 276 285 297 297 306 312 315 315 330 342 351 357 360
```

Mapa z której powstał plik: (3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45)
Znaleziona mapa: (3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45)
Czas wykonywania: 0.0140005 sekund

> test_02.txt

```
2 2 2 2 3 3 4 4 4 4 5 5 6 6 7 7 7 8 8 9 9 9 9 10 10 11 12 13 13 13 13 13
15 15 15 16 16 17 17 17 18 18 19 20 20 20 22 22 22 22 24 25 26 26 28 28 29
30 30 31 32 33 35 35 37 39
```

Mapa z której powstał plik: (4, 3, 2, 4, 4, 3, 2, 7, 6, 2, 2)
Znaleziona mapa: (2, 2, 6, 7, 2, 3, 4, 4, 2, 3, 4)
Czas wykonywania: 2.2816597 sekund

> ins-PDP-11a-asc.txt
Znaleziona mapa: (4, 6, 5, 8, 3, 9, 5, 2, 4, 7, 8, 6)
Czas wykonywania: 1.0193963 sekund

> ins-PDP-11b-asc.txt
Znaleziona mapa: (38, 74, 27, 66, 42, 15, 89, 47, 35, 13, 12, 54)
Czas wykonywania: 0.0299507 sekund

> ins-PDP-11b-desc.txt
Znaleziona mapa: (38, 74, 27, 66, 42, 15, 89, 47, 35, 13, 12, 54)
Czas wykonywania: 0.0304955 sekund

> ins-PDP-12a-asc.txt
Znaleziona mapa: (4, 6, 5, 8, 3, 9, 5, 2, 4, 7, 8, 6, 6)
Czas wykonywania: 4.9988507 sekund

> ins-PDP-13a-asc.txt
Znaleziona mapa: (3, 6, 6, 8, 7, 4, 2, 5, 9, 3, 8, 5, 6, 4)
Czas wykonywania: 40.6030012 sekund

> ins-PDP-14a-asc.txt
Znaleziona mapa: (4, 6, 5, 8, 3, 9, 5, 2, 4, 7, 8, 6, 6, 3, 5)
Czas wykonywania: 1369.6104366 sekund (22.82684061 minut)

> ins-PDP-14b-asc.txt
Znaleziona mapa: (54, 12, 13, 35, 47, 89, 15, 42, 66, 27, 74, 38, 25, 57, 79)
Czas wykonywania: 0.0120786 sekund

> ins-PDP-14b-desc.txt
Znaleziona mapa: (54, 12, 13, 35, 47, 89, 15, 42, 66, 27, 74, 38, 25, 57, 79)
Czas wykonywania: 0.0118354 sekund