Algorytmy Kombinatoryczne w Bioinformatyce

Projekt 4

Jakub Sudoł
147906

Opis projektu:

Zadaniem projektowym było napisanie programu który opiera się na mapowaniu metodą częściowego trawienia. Program wczytuje plik .txt z multizbiorem a następnie na jego podstawie tworzy mapę cięć enzymu restrykcyjnego. Program miał działać na zasadzie rekurencji czyli funkcja tworząca mapę ma odwoływać się do samej siebie w celu znalezienia optymalnego rozwiązania.

Utworzenie instancji

W własnym zakresie należało stworzyć dwie instancje o długości od 5 do 14 miejsc cięć z krokiem 3. Instancje polegały na multizbiorze zapisanym w jednej linii w pliku .txt gdzie każda z liczb jest oddzielona spacją. Przykładowa instancja w pliku .txt:

```
2 3 3 4 6 6 7 8 9 9 11 12 15 15 18
```

Nie było wymagane konkretne ułożenie liczb jednak zapis miał być inny niż rozwiązanie, żeby nie przekazywać programowi poprawnego rozwiązania.

Użyte metody i ich działanie

Wczytywanie

Wczytywanie odbywa się w odpowiedniej metodzie za pomocą standardowej biblioteki fstream i getline. Wczytany plik zapisywany jest do wektora typu int.

Przygotowywanie do tworzenia mapy

Aby przygotować się do tworzenia mapy musimy sprawdzić czy wczytana instancja ma odpowiednią długość. Wykonujemy to za pomocą wzoru z symbolem Newtona:

$$|A| = \binom{k+2}{2}.$$

Gdzie |A| to ilość elementów w multizbiorze a ‘k’ to liczba cięć. Wystarczy sprawdzić czy k jest liczbą całkowitą. Jeśli nie jest, program wyświetli stosowny komunikat i zakończy działanie. Jeśli liczba jest całkowita, zostanie wyświetlony komunikat o liczbie cięć i przewidywanej liczbie elementów w mapie. Funkcja także zapisuje w zmiennej globalnej ‘maxind’ przewidywaną ilość elementów potrzebną do późniejszych etapów. Następnie, po spełnieniu warunku, zostanie wywołana inna funkcja która służy do wyznaczenia pierwszego elementu mapy. Możemy tego dokonać odejmując od największej liczby w multizbiorze drugą największą. Różnica tego działania to pierwszy element naszej mapy więc dodajemy go do wektora

‘mapa’ oraz do wektora ‘used’. Wartość największego elementu przypisywana jest zmiennej ‘maxElement’.

Tworzenie mapy

Po spełnieniu wszystkich warunków i dodaniu pierwszego elementu do mapy, funkcja rekurencyjna rozpoczyna działanie. Dzięki temu, że jest to funkcja rekurencyjna, na samym początku znajduje się sprawdzenie czy nasza mapa została odnaleziona. Sprawdzane jest to na podstawie dwóch warunków. Pierwszy z nich sprawdza czy długość mapy zgadza się z wcześniej ustaloną wartością ‘maxind’. Jeśli się zgadza to sprawdzamy czy zawartość wektora ‘multizbior’ jest taki sam jak zawartość wektora ‘used’ co wskazuje na to, że wykorzystaliśmy wszystkie elementy. Jeśli ten warunek zostanie spełniony wypisujemy rozwiązanie oraz przerywamy funkcję.
Jeśli jednak nie znaleźliśmy jeszcze rozwiązania, przechodzimy do szukania. Na początek wybieramy kandydata do następnego elementu w mapie. Pętla ‘for’ wybiera element z multizbioru a następnie porównywana jest ilość jego wystąpień w multizbiorze i w wektorze ‘used’. Jeśli w wektorze ‘used’ jest więcej lub tyle samo wystąpień lub element który chcemy sprawdzać już został sprawdzony to pętla przechodzi do następnego wywołania i wybierany jest nowy. Jeśli element może być wykorzystany, dodaje go do mapy i do wektora ‘used’.
Następnie sprawdzane są wszystkie sumy w naszej mapie. Sprawdzany kandydat dodawany jest do zmiennej ‘suma’ a następnie za pomocą pętli sprawdzana jest wystąpienie każdej możliwej sumy dodajac elementy z mapy do ‘suma’. Sprawdzana jest ilość wystąpień sumy w multizbiorze oraz w wektorze ‘used’. Jeśli w multizbiorze występuje więcej elementów takich samych jak nasza suma niż w wektorze ‘used’, nasza suma dodawana jest do ‘used’ i sprawdzamy następną możliwą sumę. Jeśli natomiast nie znaleźliśmy sumy lub wszystkie z multizbioru zostały wykorzystane, porzucamy naszego kandydata. Usuwamy go z mapy a następnie usuwamy wszystkie elementy z wektora ‘used’ które zostały dodane na tym etapie.
Jeśli poprzedni warunek został spełniony sprawdzamy czy różnica największej sumy od największego elementu istnieje w naszym multizbiorze. Ponownie jak w poprzednich krokach liczymy wystąpienia różnicy w multizbiorze i w wektorze ‘used’. Jeśli ilość elementów znalezionych w multizbiorze jest większa od elementów znalezionych w ‘used’ oraz ilość elementów w naszej mapie jest mniejsza niż przewidywana wartość, kandydat nadaje się by został w mapie, zachodzi rekurencja i sprawdzany jest następny kandydat. Jeśli ilość elementów w mapie zgadza się z oczekiwaną, nie znaleziono żadnego wystąpienia dla powyższej różnicy i suma jest taka sama jak największy element oznacza to, że mamy kompletną mapę. Następuje rekurencja i wyświetlanie mapy. Jeśli ostatnie sprawdzenie nie będzie miało pozytywnego wyniku, mapa oraz wektor ‘used’ przywracane są do stanu początkowego przed sprawdzeniem danego kandydata. Zapisywany jest także

element który nie spełniał naszych wymogów by program mógł zacząć od następnego w multizbiorze i nie wpaść w nieskończoną pętlę.

Czas

Po tym jak mapa zostanie odnaleziona i wyświetlona a funkcja przerwie swoje działanie, pomiar czasu jest zatrzymywany i wywoływana jest funkcja która wyświetla zmierzony czas. Czas wyświetlany jest w zależności od trwania obliczeń w minutach, sekundach i mikrosekundach. Po wyświetleniu czasu program kończy swoje działanie

Testy

Testy przeprowadziłem na utworzonych przeze mnie dwóch instancjach dla każdego cięcia od 5 do 14 oraz instancjach przykładowych podanych na stronie projektu. Podany jest także czas wykonania tworzenia mapy. Testy zacznę od instancji stworzonych przeze mnie a później instancje które były podane na stronie.

Testy dla liczby cięć: 5

Test 1

Instancja: 2 3 3 4 4 6 6 7 7 8 9 10 12 12 13 15 15 16 19 19 22

Rozwiązana mapa: 3 4 2 6 4 3

Czas: 618 mikrosekund
Execution time konsoli: 0.031 sekund

Test 2

Instancja: 1 2 3 4 5 8 9 10 11 12 14 15 15 17 19 20 20 24 32 34 35

Rozwiązana mapa: 1 2 8 4 5 15

Czas: 632 mikrosekund
Execution time konsoli: 0.016 sekund

Testy dla liczby cięć: 8

Test 1

Instancja: 4 5 9 10 11 12 12 13 14 14 17 21 21 23 25 26 26 27 31 34 35 35 37 37 39 44 47 48 48 48 49 54 58 60 60 62 65 69 74 74 79 83 86 91 95

Rozwiązana mapa: 4 5 12 14 13 10 11 14 12

Czas: 534 mikrosekund

Execution time konsoli: 0.062 sekund

Test 2

Instancja: 4 5 6 7 9 10 11 12 13 14 15 16 16 17 18 20 23 23 25 25 26 27 29 31 31 32 33 36 38 38 41 41 42 47 48 49 53 54 56 58 61 65 67 72 79

Rozwiązana mapa: 7 5 11 9 6 10 13 4 14

Czas: 524 mikrosekund
Execution time konsoli: 0.047 sekund

Testy dla liczby cięć: 11

Test 1

Instancja: 3 4 5 7 8 8 9 9 10 12 12 13 13 13 13 14 15 17 17 18 20 21 23 23 24 25 25 26 26 27 28 31 31 31 34 35 36 37 37 38 39 44 44 44 45 46 48 49 51 51 54 54 57 57 58 61 62 62 64 67 69 70 71 71 74 74 75 79 80 82 82 83 86 88 95 95 99 108

Rozwiązana mapa: 9 4 13 8 10 13 14 9 3 5 7 13

Czas: 626 mikrosekund
Execution time konsoli: 0.031 sekund

Test 2

Instancja: 1 3 6 6 7 11 12 12 13 13 14 14 15 15 16 17 18 19 19 20 21 22 25 26 27 27 29 29 31 32 33 33 34 35 36 39 40 42 44 45 45 46 46 46 46 49 52 52 56 57 58 58 58 61 63 64 64 67 71 71 72 75 77 78 78 79 85 90 91 91 92 93 94 98 104 107 110 113

Rozwiązana mapa: 3 12 7 14 13 12 6 11 15 1 13 6

Czas: 278 mikrosekund
Execution time konsoli: 0.047 sekund

Testy dla liczby cięć: 14

Test 1

Instancja: 1 2 2 2 6 7 8 9 9 10 10 11 12 12 12 12 13 13 14 14 14 14 16 16 17 18 19 19 22 22 24 24 24 24 24 25 26 26 26 26 26 28 28 28 30 31 31 32 33 33 34 35 37 38 38 38 39 40 40 40 42 42 43 43 45 47 48 50 50 50 52 52 52 52 53 54 56 57 58 59 59 59 61 62 63 64 65 66 66 69 70 71 71 74 75 76 76 77 78 82 83 83 84 85 85 87 87 90 93 95 96 99 100 101 102 108 109 109 117 118

Rozwiązana mapa: 1 8 10 12 2 10 14 2 12 12 2 11 6 7 9

Czas: 618 mikrosekund
Execution time konsoli: 0.078 sekund

Test 2

Instancja: 1 1 2 4 6 9 9 10 11 12 13 13 13 14 15 15 15 15 16 17 17 19 19 20 21 21 23 24 24 24 24 25 25 28 28 30 30 32 33 34 34 34 34 36 36 37 37 40 42 43 44 45 47 47 48 49 49 49 49 51 52 52 53 60 60 61 61 62 62 62 64 64 66 68 71 73 73 75 76 77 77 77 79 81 83 84 85 86 86 88 92 94 96 96 96 97 97 98 101 103 105 107 109 110 111 113 113 117 120 121 122 126 126 127 128 130 137 141 143 147

Rozwiązana mapa: 4 15 1 1 13 10 9 15 13 15 9 12 13 11 6

Czas: 825 mikrosekund
Execution time konsoli: 0.063 sekund

Testy gotowe z strony

Test 1

Instancja: 2 3 4 4 5 5 6 6 6 7 7 8 8 9 10 11 11 11 11 12 13 13 14 14 15 15 16 16 17 18 19 19 19 20 20 21 21 22 23 23 25 25 25 26 26 27 27 27 30 30 31 31 32 32 35 35 36 36 38 38 38 40 41 42 42 43 44 46 46 49 51 52 53 57 57 61 63 67

Rozwiązana mapa: 4 6 5 8 3 9 5 2 4 7 8 6

Czas: 406 mikrosekund
Execution time konsoli: 0.078 sekund

Test 2

Instancja: 2 3 4 4 5 5 6 6 6 6 7 7 8 8 9 10 11 11 11 11 12 12 13 13 14 14 15 15 16 16 17 18 19 19 19 20 20 20 21 21 22 23 23 25 25 25 26 26 27 27 27 27 30 30 31 31 31 32 32 33 35 35 36 36 38 38 38 38 40 41 42 42 43 44 46 46 47 49 50 51 52 53 57 57 58 61 63 63 67 69 73

Rozwiązana mapa: 4 6 5 8 3 9 5 2 4 7 8 6 6

Czas: 750 mikrosekund
Execution time konsoli: 0.234 sekund

Test 3

Instancja: 2 3 3 4 4 5 5 6 6 6 6 7 7 8 8 9 9 10 11 11 11 11 12 12 13 13 14 14 15 15 15 16 16 17 18 19 19 19 20 20 20 21 21 22 23 23 23 25 25 25 26 26 27 27 27 27 30 30 30 31 31 31 32 32 33 34 35 35 36 36 36 38 38 38 38 40 41 41 42 42 43 44 46 46 47 49 50 50 51 52 53 53 57 57 58 61 61 63 63 66 67 69 72 73 76

Rozwiązana mapa: 3 6 6 8 7 4 2 5 9 3 8 5 6 4

Czas: 957 mikrosekund
Execution time konsoli: 0.344 sekund

Test 4

Instancja: 2 3 3 4 4 5 5 5 6 6 6 6 7 7 8 8 8 9 9 10 11 11 11 11 12 12 13 13 14 14 14 15 15 15 16 16 17 18 19 19 19 20 20 20 20 21 21 22 23 23 23 25 25 25 26 26 27 27 27 27 28 30 30 30 31 31 31 32 32 33 34 35 35 35 36 36 36 38 38 38 38 39 40 41 41 41 42 42 43 44 46 46 46 47 49 50 50 51 52 53 53 55 57 57 58 58 61 61 63 63 66 66 67 69 71 72 73 76 77 81

Rozwiązana mapa: 4 6 5 8 3 9 5 2 4 7 8 6 6 3 5

Czas: 3 sekundy 433 mikrosekund
Execution time konsoli: 3.960 sekund

Test 5

Instancja: 12 13 15 25 27 35 38 42 47 48 54 57 60 66 66 74 79 82 89 93 95 101 104 107 108 112 114 123 135 136 139 146 150 151 161 167 171 184 186 193 196 199 205 209 211 212 224 228 239 241 247 250 253 259 262 265 286 294 307 307 313 319 321 334 346 351 360 373 395 398 400 408 420 433 446 458 474 512

Rozwiązana mapa: 38 74 27 66 42 15 89 47 35 13 12 54

Czas: 617 mikrosekund
Execution time konsoli: 0.047 sekund

Test 6

Instancja: 512 474 458 446 433 420 408 400 398 395 373 360 351 346 334 321 319 313 307 307 294 286 265 262 259 253 250 247 241 239 228 224 212 211 209 205 199 196 193 186 184 171 167 161 151 150 146 139 136 135 123 114 112 108 107 104 101 95 93 89 82 79 74 66 66 60 57 54 48 47 42 38 35 27 25 15 13 12

Rozwiązana mapa: 38 74 27 66 42 15 89 47 35 13 12 54

Czas: 625 mikrosekund
Execution time konsoli: 0.047 sekund

Test 7

Instancja: 12 13 15 25 25 27 35 38 42 47 48 54 57 57 60 63 66 66 74 79 79 82 82 89 93 95 101 104 107 108 112 114 120 123 135 136 136 137 139 146 150 151 161 161 164 167 171 184 186 193 194 196 199 199 205 209 211 212 221 224 228 230 239 241 247 250 253 259 262 265 272 273 286 287 287 294 300 307 307 313 319 321 329 334 344 346 351 360 366 373 376 395 398 400 408 408 420 423 423 433 433 446 458 458 471 474 480 483 512 512 515 528 537 540 559 594 594 607 619 673

Rozwiązana mapa: 54 12 13 35 47 89 15 42 66 27 74 38 25 57 79

Czas: 254 mikrosekund
Execution time konsoli: 0.034 sekund

Test 8

Instancja: 673 619 607 594 594 559 540 537 528 515 512 512 483 480 474 471 458 458 446 433 433 423 423 420 408 408 400 398 395 376 373 366 360 351 346 344 334 329 321 319 313 307 307 300 294 287 287 286 273 272 265 262 259 253 250 247 241 239 230 228 224 221 212 211 209 205 199 199 196 194 193 186 184 171 167 164 161 161 151 150 146 139 137 136 136 135 123 120 114 112 108 107 104 101 95 93 89 82 82 79 79 74 66 66 63 60 57 57 54 48 47 42 38 35 27 25 25 15 13 12

Rozwiązana mapa: 54 12 13 35 47 89 15 42 66 27 74 38 25 57 79

Czas: 362 mikrosekund
Execution time konsoli: 0.060 sekund

Wnioski

Program ze względu na zastosowanie funkcji rekurencyjnej nie należał do najłatwiejszych. Trzeba było znaleźć takie rozwiązanie problemu które nie było zbyt obciążające dla komputera. Wykorzystanie wektora z zużytymi zawartościami zdało o wiele lepszy egzamin niż moja wcześniejsza próba usuwania na bieżąco z multizbioru i przechowywanie poprzednich wersji w wektorze wektorów ze wzgledu na błąd stack overflow. Wyniki testów są mocno zróżnicowane ponieważ nie tylko wielkość multizbioru ma znaczenie ale także łatwość w znalezieniu dobrego rozwiązania czy aktualne obciążenie procesora przez co testy które mają więcej elementów wcale nie muszą wykonywać się dłużej niż te z mniejszą ilością.