

NOT: Bu adimlar algoritmanin nasil calistigini daha iyi anlamak icin extra olarak analiz yaparken incelenmistir.

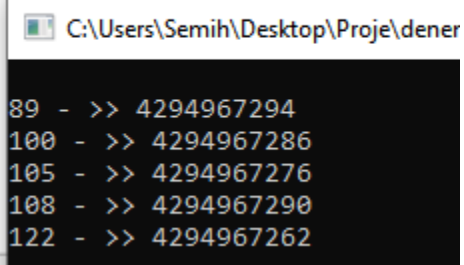
Yildiz patterni icin bir deneme

Pattern = "Yildiz"

Text = "merhaba mehmet semih babacan Yildiz teknik unviuersitesi ogrencisi olmaktan dolayi gurur duyuyor"

```
>>> for i, item in zip([89, 100, 105, 108, 122],a):
    print(i, "->", bin(item))

89 -> 0b11111111111111111111111111111110
100 -> 0b111111111111111111111111111110110
105 -> 0b111111111111111111111111111101100
108 -> 0b11111111111111111111111111111010
122 -> 0b11111111111111111111111111011110
>>>
```



C:\Users\Semih\Desktop\Proje\dener

```
89 - >> 4294967294
100 - >> 4294967286
105 - >> 4294967276
108 - >> 4294967290
122 - >> 4294967262
```

Y harfi text[89] > 0b11111111111111111111111111111110

l harfi text[100] > 0b111111111111111111111111111110110

d harfi text[105] > 0b111111111111111111111111111101100

i harfi text[108] > 0b11111111111111111111111111111010

z harfi text[122] > 0b11111111111111111111111111011110

Asagidaki gibi bir vector dizisi olusturulacak her bir harf icin verilen degerleri goruyoruz

0. m binarysi

0b111111111111111111111111111111100

1. e binarysi

0b111111111111111111111111111111100

2. r binarysi

0b111111111111111111111111111111100

3. h binarysi

0b111111111111111111111111111111100

4. a binarysi

0b111111111111111111111111111111100

5. b binarysi

6. a binarysi

```
0b1111111111111111111111111111111111111111111111111111111
```

7. binarysi

0b11111111111111111111111111111111100

8. m binarysi

[illegible]

9. e binarysi

```
0b11111111111111111111111111111111100
```

10. h binarysi

[illegible]

11. m binarysi

[illegible]

12. e binarysi

```
0b11111111111111111111111111111111100
```

13. t binarysi

```
0b11111111111111111111111111111111100
```

14. binarysi

0b11111111111111111111111111111111100

15. s binarysi

[illegible]

16. e binarysi

```
0b11111111111111111111111111111111
```

17. m binarysi

[illegible]

18. i binarysi

```
0b11111111111111111111111111111111000
```

19. h binarysi

0b11111111111111111111111111111111100

20. binarysi

[illegible]

21. b binarysi

```
0b11111111111111111111111111111111100
```

22. a binarysi

```
0b1111111111111111111111111111111100
```

23. b binarysi

[illegible]

24. a binarysi

```
0b11111111111111111111111111111111100
```

25. c binarysi

0b11111111111111111111111111111111100

26. a binarysi

```
0b11111111111111111111111111111111100
```

27. n binarysi

0b11111111111111111111111111111111100

28. binarysi

```
0b11111111111111111111111111111111100
```

29. Y binarysi

```
0b11111111111111111111111111111111100
```

30. i binarysi

0b111111111111111111111111111111111000

31. I binarysi

[illegible]

32. d binarysi

```
0b11111111111111111111111111111111101100
```

33. i binarysi

0b111111111111111111111111111111111011000

34. z binarysi

49. s binarysi

Ob1111111111111111111111111111111100

50. i binarysi

```
0b11111111111111111111111111111111000
```

51. t binarysi

```
0b1111111111111111111111111111111100
```

52. e binarysi

[illegible]

53. s binarysi

```
0b1111111111111111111111111111111100
```

54. i binarysi

```
0b11111111111111111111111111111111000
```

55. binarysi

```
0b1111111111111111111111111111111100
```

56. o binarysi

0b11111111111111111111111111111111100

57. g binarysi

0b11111111111111111111111111111111100

58. r binarysi

```
0b1111111111111111111111111111111100
```

59. e binarysi

0b11111111111111111111111111111111100

60. n binarysi

0b11111111111111111111111111111111100

61. c binarysi

0b11111111111111111111111111111111100

62. i binarysi

0b11111111111111111111111111111111000

63. s binarysi

78. a binarysi

0b11111111111111111111111111111100

79. y binarysi

0b11111111111111111111111111111100

80. i binarysi

0b11111111111111111111111111111000

81. binarysi

0b11111111111111111111111111111100

82. g binarysi

0b11111111111111111111111111111100

83. u binarysi

0b11111111111111111111111111111100

84. r binarysi

0b11111111111111111111111111111100

85. u binarysi

0b11111111111111111111111111111100

86. r binarysi

0b11111111111111111111111111111100

87. binarysi

0b11111111111111111111111111111100

88. d binarysi

0b11111111111111111111111111111100

89. u binarysi

0b11111111111111111111111111111100

90. y binarysi

0b11111111111111111111111111111100

91. u binarysi

0b11111111111111111111111111111100

92. y binarysi

Kontrolumuz dogru cikiyor.

Tipki asagidaki matris yapisi gibi bir matris hayal edebiliriz

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
		G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G
0	G	0	1	1	1	1	0	1	1	0	1	0	1	0	1	1	1	1	1	0	1	1	1	1	0
1	C	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	A	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	G	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	A	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	G	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
6	A	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
7	G	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

Her bir R degeri icin verilen bit vektoru seklinde degerler

0. R -> binary

0b111111111111111111111111111100

1. R -> binary

0b111111111111111111111111111100

2. R -> binary

0b111111111111111111111111111100

3. R -> binary

0b111111111111111111111111111100

4. R -> binary

0b111111111111111111111111111100

5. R -> binary

0b111111111111111111111111111100

6. R -> binary

0b111111111111111111111111111100

7. R -> binary

0b111111111111111111111111111100

8. R -> binary

0b111111111111111111111111111100

9. R -> binary

Ob1111111111111111111111111111111100

10. R -> binary

0b11111111111111111111111111111111100

11. R -> binary

```
0b11111111111111111111111111111111
```

12. R -> binary

[illegible]

13. R -> binary

```
0b11111111111111111111111111111111100
```

14. R -> binary

0b11111111111111111111111111111111100

15. R -> binary

```
0b1111111111111111111111111111111100
```

16. R -> binary

0b11111111111111111111111111111111100

17. R -> binary

```
0b11111111111111111111111111111111100
```

18. R -> binary

```
0b11111111111111111111111111111111000
```

19. R -> binary

0b11111111111111111111111111111111100

20. R -> binary

0b11111111111111111111111111111111100

21. R -> binary

0b11111111111111111111111111111111100

22. R -> binary

```
0b1111111111111111111111111111111100
```

23. R -> binary

38. R -> binary

Ob1111111111111111111111111111111100

39. R -> binary

[illegible]

40. R -> binary

```
0b11111111111111111111111111111111
```

41. R -> binary

[illegible]

42. R -> binary

```
0b11111111111111111111111111111111100
```

43. R -> binary

[illegible]

44. R -> binary

[illegible]

45. R -> binary

[illegible]

46. R -> binary

[illegible]

47. R -> binary

[illegible]

48. R -> binary

```
0b11111111111111111111111111111111100
```

49. R -> binary

[illegible]

50. R -> binary

[illegible]

51. R -> binary

```
0b1111111111111111111111111111111100
```

52. R -> binary

[illegible]

53. R -> binary

[illegible]

54. R -> binary

0b11111111111111111111111111111111000

55. R -> binary

[illegible]

56. R -> binary

```
0b1111111111111111111111111111111100
```

57. R -> binary

`0b11111111111111111111111111111111100`

58. R -> binary

[illegible]

59. R -> binary

0b11111111111111111111111111111111100

60. R -> binary

```
0b11111111111111111111111111111111100
```

61. R -> binary

[illegible]

62. R -> binary

```
0b11111111111111111111111111111111000
```

63. R -> binary

0b11111111111111111111111111111111100

64. R -> binary

[illegible]

65. R -> binary

0b11111111111111111111111111111111100

66. R -> binary

0b11111111111111111111111111111111100

67. R -> binary

0b1111111111111111111111111111111100

68. R -> binary

```
0b11111111111111111111111111111111100
```

69. R -> binary

```
0b1111111111111111111111111111111100
```

70. R -> binary

[illegible]

71. R -> binary

```
0b11111111111111111111111111111111100
```

72. R -> binary

[illegible]

73. R -> binary

[illegible]

74. R -> binary

[illegible]

75. R -> binary

[illegible]

76. R -> binary

[illegible]

77. R -> binary

```
0b11111111111111111111111111111111100
```

78. R -> binary

[illegible]

79. R -> binary

[illegible]

80. R -> binary

```
0b11111111111111111111111111111111000
```

81. R -> binary

ACIKLAMA

34. i degerinde 1111111111111111111111110111100

ile bit gosterimiyle 1000000 yapilan islem sonucu

```
if (0 == (R & (1UL << n)))
```

```
return (i - n) + 1;
```

1111111111111111111111110111100 ->>> R icin

00000000000000000000000000000000 - >>> R ile karsilastirilan ($1UL \ll n$) degeri

Sonuc 0 olur ve if için dogru bir kosul saglanir. $(i - n) + 1$ degeri de return edilmiş olur.