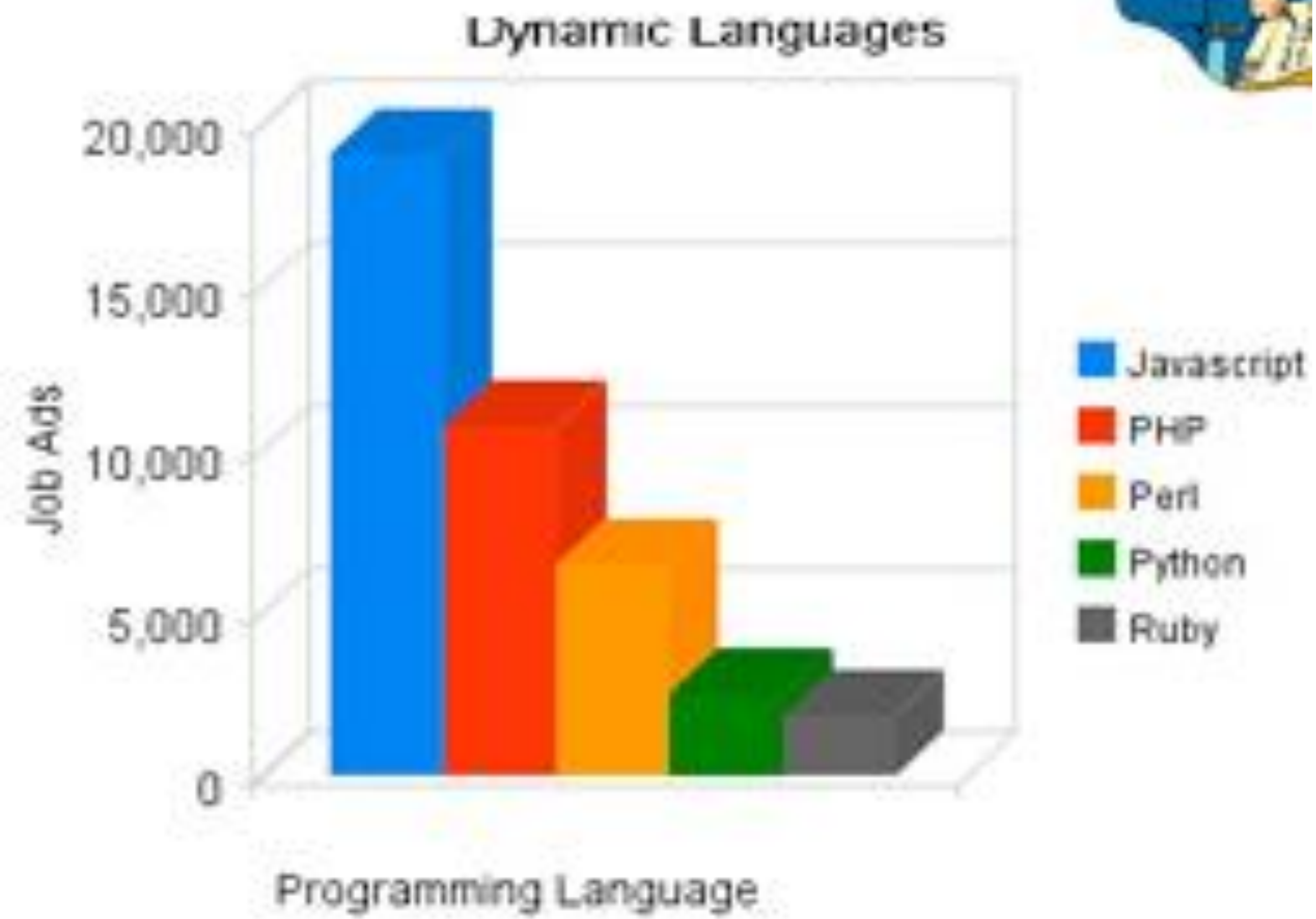
SNS College of Engineering
Coimbatore - 641107

Computing Binomial Co-efficients

Dynamic Programming

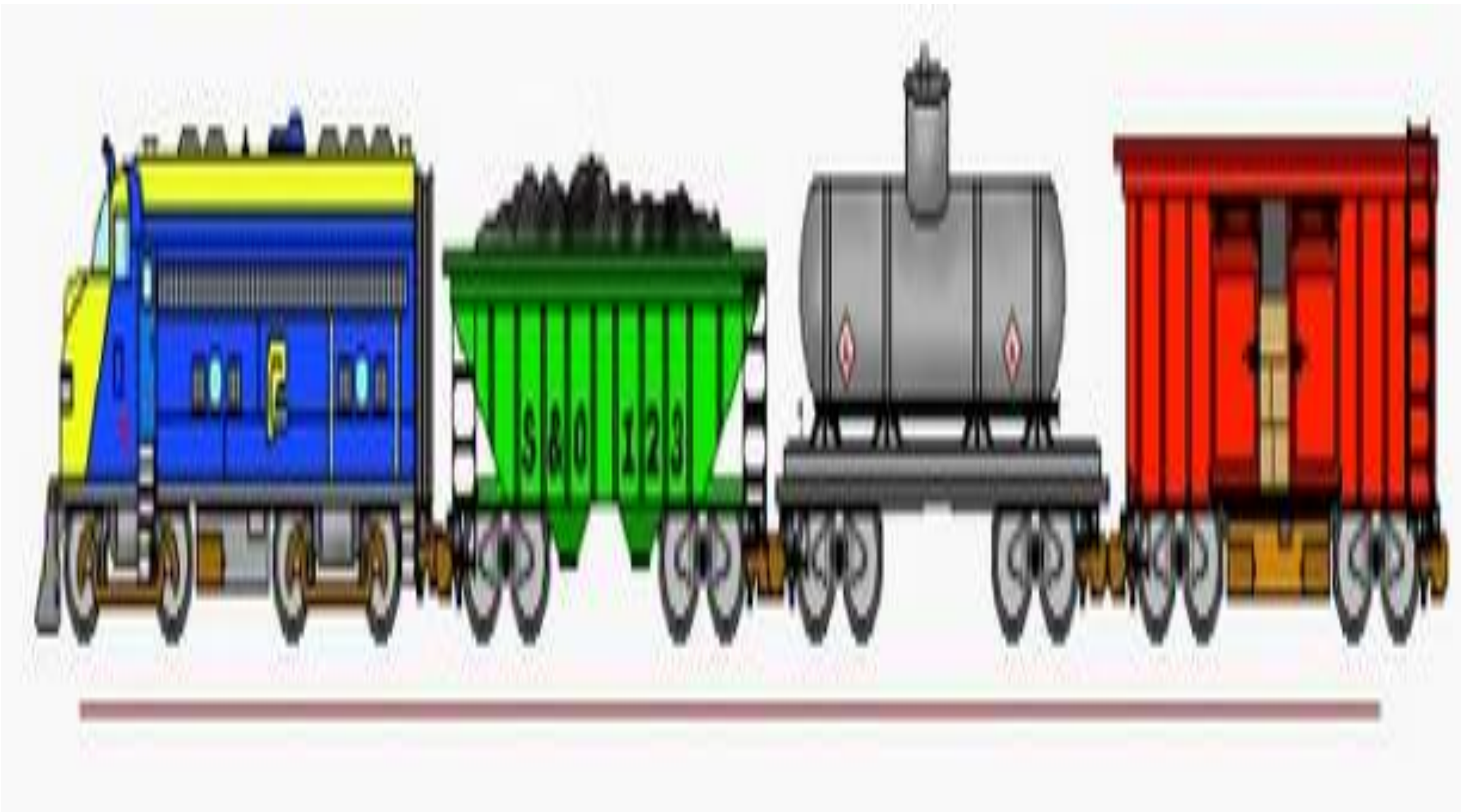
- Dynamic – changing
- Programming- Planning


Dynamic Languages
20,000
15,000
10,000
5,000
0
Job Ads
Programming Language
Javascript
PHP
Perl
Python
Ruby

Overlapping the sub problems

Principle of optimality

sequence of decision making



Computing binomial coefficients

- Computing binomial coefficients is non optimization problem but can be solved using dynamic programming.

- Binomial coefficients are represented by $C(n, k)$ or $\binom{n}{k}$ and can be used to represent the coefficients of a binomail:

$$(a + b)^n = C(n, 0)a^n + \dots + C(n, k)a^{n-kbk} + \dots + C(n, n)b^n$$

- The recursive relation is defined by the prior power

$$C(n, k) = C(n-1, k-1) + C(n-1, k) \text{ for } n > k > 0$$

$$\text{IC } C(n, 0) = C(n, n) = 1$$

$N*k$ table

		k					
		0	1	2	…	k-1	k
	0	1					
	1	1	1				
	2	1	2	1			
n	.						
	.						
	.						
	k	1					1
	.						
	.						
	.						
	n-1	1				$C(n\text{-}1, k\text{-}1)$	
	n	1					$C(n, k)$

Algorithm

Algorithm *Binomial*(*n*, *k*)

for *i* ← 0 **to** *n* **do** // fill out the table row wise

for *i* = 0 **to** min(*i*, *k*) **do**

if *j*==0 or *j*==*i* **then** *C*[*i*, *j*] ← 1 // IC

else *C*[*i*, *j*] ← *C*[*i*-1, *j*-1] + *C*[*i*-1, *j*] // recursive relation

return *C*[*n*, *k*]

Analysis

$A(n, k)$ = sum for upper triangle + sum for the lower rectangle

$= \sum_{i=1}^{k} \sum_{j=1}^{i-1} 1 + \sum_{i=1}^{n} \sum_{j=1}^{k} 1$

$= \sum_{i=1}^{k} (i-1) + \sum_{i=1}^{n} k$

$= (k-1)k/2 + k(n-k) \in \Theta(nk)$

Activity

```
                              C(5, 2)
                          /             \
              C(4, 1)                        C(4, 2)
               /  \                         /       \
      C(3, 0)   C(3, 1)           C(3, 1)              C(3, 2)
                 /    \            /    \              /    \
C(2, 0)    C(2, 1)      C(2, 0) C(2, 1)          C(2, 1)  C(2, 2)
              /       \             /  \           /   \
     C(1, 0)  C(1, 1)     C(1, 0)  C(1, 1)  C(1, 0)  C(1, 1)
```