# Programação Dinâmica

Prof. Igor Natal
Dep. De Informática
UEM

# Introdução

Programação Dinâmica é uma técnica de otimização que transforma um problema complexo em uma sequência de problemas mais simples (se for possível!). Sua característica essencial é a estrutura de problemas de otimização em múltiplos estágios, que são resolvidos sequencialmente. Cada estágio é resolvido como um problema de otimização, e suas soluções ajudam a definir as características dos próximos estágios na sequência.

# Introdução

- Alguns chamam de resolução recursiva com memória.
- Programação Dinâmica pode ser entendida como um paradigma de modelagem de problema e implementação de algoritmo.
- Ao contrário da programação linear, programação dinâmica não possui um padrão de modelagem. Programação dinâmica pode utilizar de tabela, grafos, diagramas, etc.

# Introdução

Histórico:

Wald (1950) - decisão sequencial

Dvoretzky, Kiefer & Wolfowitz (1952) - estoques

Bellman (década de 50) - artigos, livros.

Ideia:

obter uma série de problemas com um único estágio e um número menor de variáveis.

# Princípio de Bellman para Programação Dinâmica

"Dado o estado atual, uma política ótima para as etapas restantes é independente da política adotada em etapas anterior. Portanto, a decisão ótima para a etapa atual depende somente do estado atual".

Ou seja: "Uma política de decisões ótimas só pode ser formada por subpolíticas ótimas"

Bibliografia complementar sugerida:

Marco Cesar Goldbarg, Henrique Pacca. **Otimização Combinatória e Programação Linear**. 2005. Editora Campus.

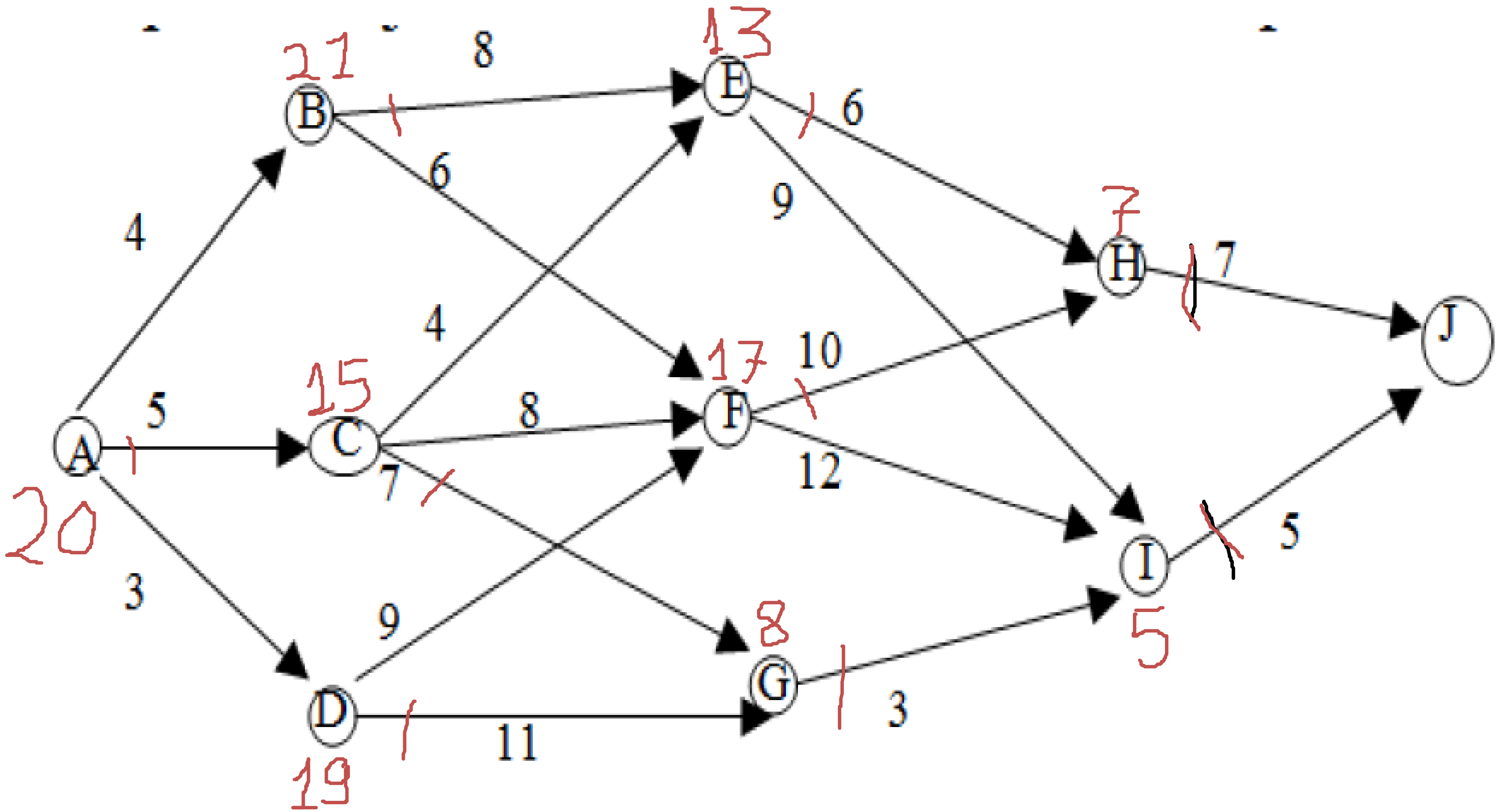
# Terminologias da Programação Dinâmica

- Estado: uma configuração do sistema;
- Estágio ou etapa: é uma fase de transição. Muitas vezes relacionado ao tempo;
- Ação ou decisão: para cada estado deve ser selecionada uma decisão;
- Política: uma decisão (ação) tomada em um estado;
- Plano: uma sequência de decisões tomadas em cada etapa;
- Retorno: é algo que o sistema gera sobre uma etapa. Ex. Lucro, custo, distância, tempo gasto, etc.
- Valor do estado: é uma função matemática baseada nos retornos de cada estágio. Estes valores compõem a função objetivo do problema.

# Problema da Viagem

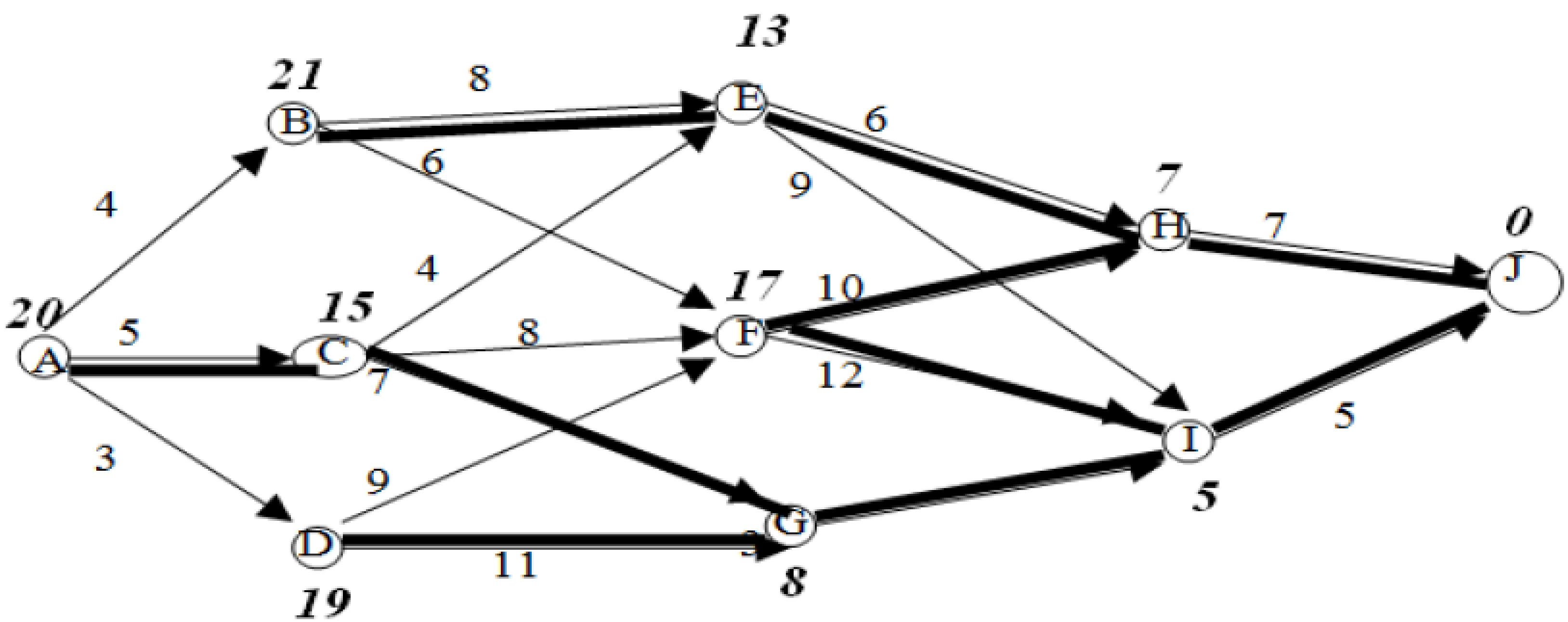
Viajante sai da cidade A com destino a cidade J.

Qual a trajetória de menor custo de transporte?

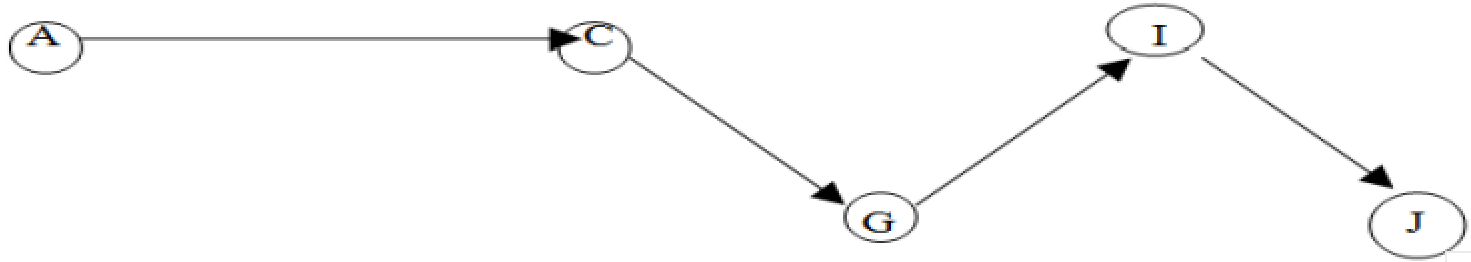


# Problema da Viagem

Solução sobre o Grafo, partindo da Cidade Destino J e considerando o **Custo Remanescente Mínimo** de cada Cidade até J .




**Custo mínimo da viagem entre as cidades A e J= 20**



# Elementos de PD para o Menor Caminho em Grafo

Estágios: (instante de início/término de uma viagem)

Estado: (cidade onde o viajante está num dado estágio - exemplo: no penúltimo estágio há os estados *h* e *i*)

Decisão: (para ir de uma cidade à outra - de outro estágio - o viajante deve tomar uma decisão admissível)

Custos (elementares): da tomada de decisão (quanto custa para ir de uma cidade à outra)

Plano: (a trajetórias de menor custo para o viajante)

# Elementos importantes em modelos de PD: ilustração com o Problema da Viagem

Estilos de resolução

- ***Backward***: fixar o último estado = objetivo estabelecido - uso em operação de sistemas.
- ***Forward***: fixar o estado inicial = condições iniciais conhecidas - uso em planejamento.

Recuperação da trajetória - percorrer o grafo no sentido contrário (***Backtracking***) utilizado para os cálculos efetuados (obter as cidades intermediárias que compõem a trajetória ótima até a Cidade J a partir da Cidade A).

# Elementos da Programação Dinâmica

O problema pode ser dividido em *estágios* (ou etapas), que requerem uma política de decisão em cada uma delas. Denotemos o estágio por n. Frequentemente, os estágios representam períodos de tempo no horizonte planejamento do problemas. Cada estágio tem um certo número de estados associado. Denotemos cada estado pelo par ordenado $(n, i)$.

Dado um *estado* $(n,i)$ e se for tomado uma *decisão* $k$ então o estado no próximo estágio é $(n+1, j)$, onde $j$ é obtido pela função de transição de transição $T(n,i,k)$, isto é, $T(n,i,k)=j$.

O *valor de um estado* é pela obtido pela seguinte **função recursiva**: $f(n,i) = \min_k \{r(n,i,k)+f(n+1,j)\}$, onde $r(n,i,k)$ é o custo (ou retorno) para o estado $(n, i)$ se a decisão $k$ for tomada. O valor pode ser entendido como o custo imediato (estágio $n$) + o custo futuro (estágios $n$+1 em diante). A função de recursiva pode ser de maximização se o problema desejar maximizar o ganho (ou retorno).

Dado o estado atual, uma *política ótima* para as etapas restantes é independente da política adotada em etapas anteriores. Portanto, a decisão ótima atual depende somente do estado atual e de como se chegou ali. Este é o **princípio de otimalidade** para a programação dinâmica.

# Programação Dinâmica

Como resolver um problema de programação dinâmica?

- Diferente da Prog. Linear que os modelos tem uma estrutura bem definida (função objetivo, restrições), em Programação Dinâmica (PD) é comum utilizar grafos representativos ou tabelas.
- Adotaremos o uso de grafos nesta disciplina.

# Análise de Complexidade

Considere:

$|A|$ o número de decisões;

$|S|$ o número de estados;

$n$ o número de estágios;

Enumeração Exaustiva: $O(2^{|A|})$

Programação Dinâmica: $O(n\,|S|\,|A|)$

Geralmente PD tende a ser mais eficiente do que uso de recursividade.

# Exercício 1

## Problema de Planejamento de Estudo

Um aluno está prestes a iniciar seu período de exames em três disciplinas, sendo que ele tem 3 dias disponíveis para sua preparação. Além disso, o aluno durante um dia só estuda para uma disciplina, por uma questão de método, e quer estar presente em todos os exames.

A previsão do aluno para a nota em cada uma das disciplinas, em função do tempo (dias) de preparação para uma delas, é a seguinte:

| Dias \ Disciplina | A | B | C |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 3 | 4 | 5 |
| 2 | 7 | 8 | 9 |
| 3 | 8 | 9 | 10 |

O aluno pretende saber qual o plano de estudo (dias de estudos) que maximizará a média das notas nos exames. Formule o problema.

# Exemplo 2

Problema de Planejamento de Estudo

Para efeito de comparação de técnicas, vamos modelar este problema de duas formas:

1. Por programação linear;

2. Por programação dinâmica.

Descisão $k$: Quantos dias dedicar para a disciplina

Estágios (camadas): disc. A, disc. B, disc. C e "FIM".

Estado *(vértice)*: número de dias disponíveis para estudo

Retorno $r$: nota esperada na disciplina em função de $k$.

# Exercício 2

## PROBLEMA DA CONSTRUÇÃO DE ESTRADAS

Uma empresa planeja construir 3 trechos de estadas A, B e C, um por ano e pode escolher a ordem que serão construídos. Além disso, em função das características dos trechos, os custos dependem da ordem que serão construídos segundo a tabela abaixo.

| Já construídos | Custo | | |
|---|---|---|---|
| | A | B | C |
| - | 10 | 8 | 6 |
| A | - | 9 | 8 |
| B | 13 | - | 9 |
| C | 11 | 10 | - |
| A, B | - | - | 11 |
| A, C | - | 12 | - |
| B, C | 14 | - | - |

Determine a sequência de obras que minimize os custos.

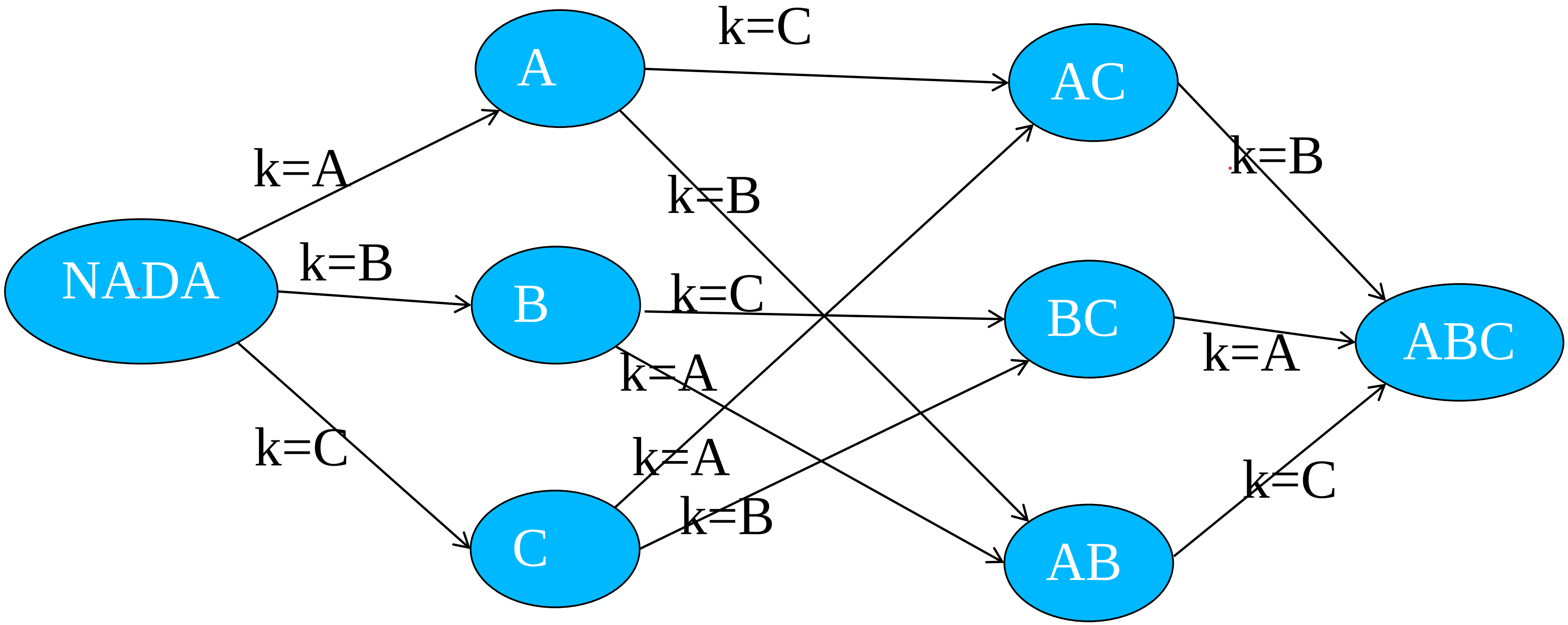
# Exercício 2 - Resolução

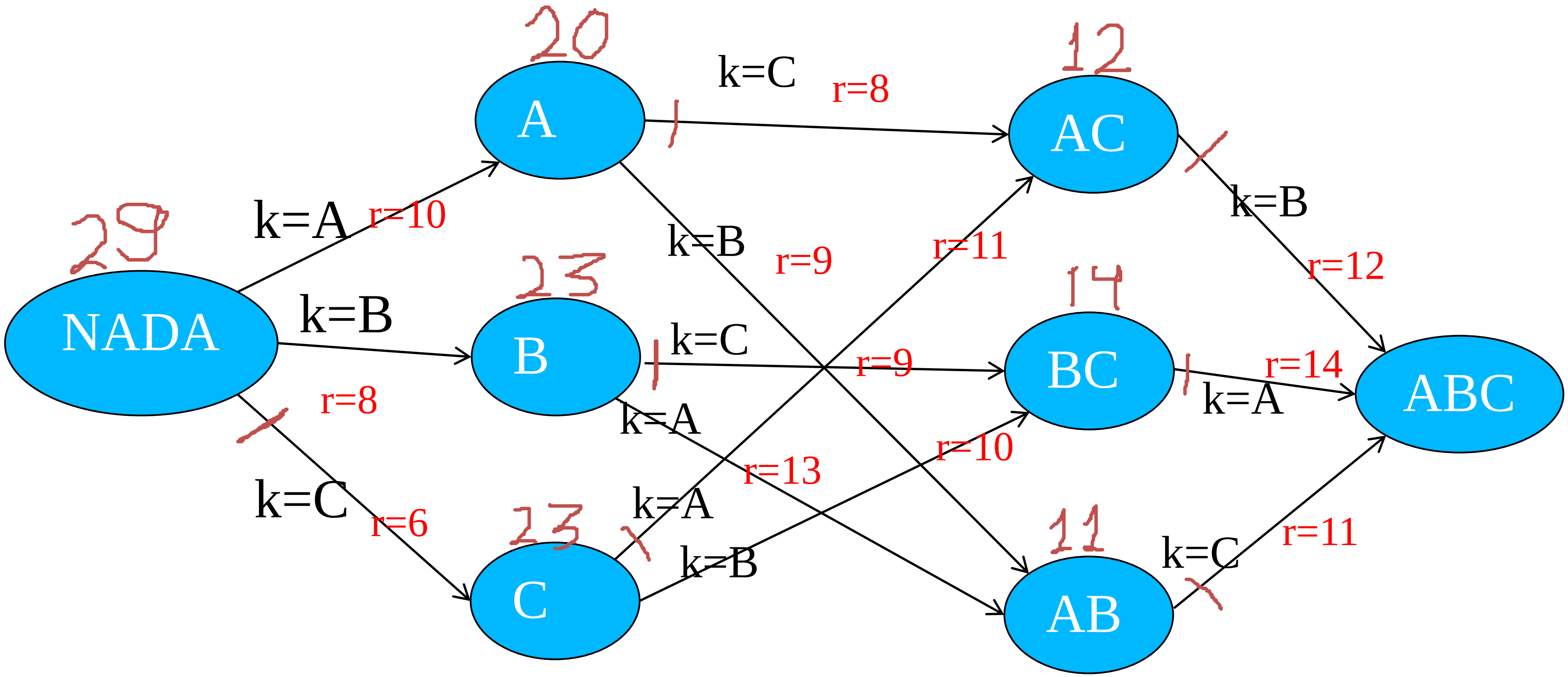
Descisão *k*: trecho a ser construído

Estágios (camadas): ano de execução

Estado (vértice): trecho já construído

Retorno *r*:  custo  de construção


k=C
A
AC
k=A
k=B
k=B
NADA
k=B
B
k=C
BC
ABC
k=A
k=A
k=C
k=A
k=C
k=B
C
AB




Política ótima (Ordem de construção): C, A, B

# Exercício 3

**PROBLEMA DA MOCHILA (BINÁRIO)**

# Exercício 3 - Resolução

Descisão *k*: se o item entra ou não na mochila

Estágios (camadas): itens da mochila (4 itens).

Estado (vértice): espaço disponível na mochila

Retorno *r*: lucro ao inserir o item na mochila

# Exercício 6

**PROBLEMA DE SOBREVIVÊNCIA**

Considerar a probabilidade de estar vivo.

Probabilidade condicionada: multiplicar as probabilidades.

# Exercício 4

DISTRIBUIÇÃO DE CIENTISTAS EM EQUIPES DE PESQUISA

Considerar a probabilidade de estar vivo.

Probabilidade condicionada: multiplicar as probabilidades.

# Exercício 4 - Resolução

Descisão *k*: Quantos cientistas adicionais a serem alocados a equipe

Estágios (camadas): corresponde as equipes (1,2,3)

Estado (vértice): Número de novos cientistas ainda disponível

Retorno *r*: probabilidade de falha