A Tabela 1.1 apresenta uma comparação entre os algoritmos dos Programas 1.2, 1.3 e 1.4,. considerando o número de comparações como medida de complexidade. Os algoritmos MaxMin2 e MaxMin3 são superiores ao algoritmo MaxMinl de forma geral. O algoritmo MaxMin3 é superior ao algoritmo MaxMin2 com relação ao pior caso e bastante próximo quanto ao caso médio.

Os Três Algoritmos	$f(n)$		
	Melhor caso	Pior caso	Caso médio
MaxMin1	2(n-1)	2(n-1)	2(n-1)
MaxMin2	$n-1$	2(n-1)	$3n/2-3/2$
MaxMin3	$3n/2-2$	$3n/2-2$	$3n/2-2$

Tabela 1.1: Comparação dos algoritmos para obter o máximo e o mínimo

Considerando novamente o número de comparações realizadas, existe possibilidade de obter um algoritmo mais eficiente para este problema? Para responder a esta questão é necessário conhecer o **limite inferior** para a classe de algoritmos para obter o maior e o menor elemento de um conjunto.

Uma técnica muito utilizada para obter o limite inferior para uma classe qualquer de algoritmos é através da utilização de um oráculo.[2] Dado um modelo de computação que expresse o comportamento do algoritmo o oráculo informa o resultado de cada passo possível, que no nosso caso seria o resultado de cada comparação. Para derivar o **limite inferior** o oráculo procura sempre fazer com que o algoritmo trabalhe o máximo, escolhendo como resultado da próxima comparação aquele que cause o maior trabalho possível que é necessário para determinar a resposta final.

O teorema abaixo, apresentado por Horowitz e Sahni (1978, p.476), utiliza um oráculo para derivar o limite inferior no número de comparações necessárias para obter o máximo e o mínimo de um conjunto com n elementos.

Teorema: Qualquer algoritmo para encontrar o maior elemento e o menor elemento de um conjunto com n elementos não ordenados, $n \geq 1$, faz pelo menos $[3n/2] - 2$ comparações.

Prova: A técnica utilizada define um oráculo que descreve o comportamento do algoritmo através de um conjunto de n–tuplas, mais um conjunto de regras associadas que mostram as tuplas possíveis (estados) que um algoritmo pode assumir a partir de uma dada tupla e uma única comparação.

[2]De acordo com o Novo Dicionário Aurélio da Lingua Portuguesa, um oráculo é: 1. Resposta de um deus a quem o consultava. 2. Divindade que responde consultas e orienta o crente: o oráculo de Delfos. 3. Fig. Palavra, sentença ou decisão inspirada, infalível ou que tem grande autoridade: os oráculos dos profetas, os oráculos da ciência.

O comportamento do algoritmo pode ser descrito por uma 4-tupla, representada por *(a, b, c, d),* onde a representa o número de elementos que nunca foram comparados; *b* representa o número de elementos que foram vencedores e nunca perderam em comparações realizadas; c representa o número de elementos que foram perdedores e nunca venceram em comparações realizadas; *d* representa o número de elementos que foram vencedores e perdedores em comparações realizadas. O algoritmo inicia no estado *(n,* 0, *0, 0)* e termina com (0, 1,1, *n — 2).* Desta forma, após cada comparação a tupla *(a, b, c, d)* consegue progredir apenas se ela assume um dentre os cinco estados possíveis, a saber:

$(a-2, b+1, c+1, d)$	se $a \geq 2$	{ dois elementos de a são comparados }
$(a-1, b+1, c, d)$ ou		
$(a-1, b, c+1, d)$	se $a \geq 1$	{ um elemento de a comparado com um de b ou um de c }
$(a, b-1, c, d+1)$	se $b \geq 2$	{ dois elementos de b são comparados }
$(a, b, c-1, d+1)$	se $c \geq 2$	{ dois elementos de c são comparados }

O primeiro passo requer necessariamente a manipulação do componente a. Observe que o caminho mais rápido para levar o componente a até zero requer $\lceil n/2 \rceil$ mudanças de estado e termina com a tupla $(0, n/2, n/2, 0)$, através da comparação dos elementos de a dois a dois. A seguir, para reduzir o componente b até um são necessárias $\lceil n/2 \rceil - 1$ mudanças de estado, correspondente ao número mínimo de comparações que é necessário para obter o maior elemento de b. Idem para c, com $\lceil n/2 \rceil - 1$ mudanças de estado. Logo, para obter o estado $(0, 1, 1, n-2)$ a partir do estado $(n, 0, 0, 0)$ são necessárias

$$\lceil n/2 \rceil + \lceil n/2 \rceil - 1 + \lceil n/2 \rceil - 1 = \lceil 3n/2 \rceil - 2$$

comparações. □

O teorema acima nos diz que se o número de comparações entre os elementos de um vetor for utilizado como medida de custo então o algoritmo MaxMin3 do Programa 1.4 é **ótimo.**

1.3.1 Comportamento Assintótico de Funções

Como já foi observado anteriormente, o custo para obter uma solução para um dado problema aumenta cem o tamanho *n* do problema. O número de comparações para encontrar o maior elemento de um conjunto de *n* inteiros, ou para ordenar os elementos de um conjunto com *n* elementos, aumenta com *n:* 0 parâmetro *n* fornece uma medida da dificuldade para se resolver