**RIT Case Brief – ALGO 1**
**Build 1.02**

# Algorithmic Arbitrage 1

This case is designed to introduce students to algorithmic trading by providing a simple example of exploiting an arbitrage opportunity for one stock traded on two different exchanges.

CRZY shares trade on both the "Main" (M) exchange and the "Alternate" (A) exchange. Each exchange maintains an independent limit order book for the stock, and orders can be routed to either exchange. Since the same stock is being traded on both exchanges, the positions are aggregated. If one were to purchase 100 shares of CRZY on the main exchange, they can immediately sell 100 shares of CRZY on the alternate exchange and the result is a zero-position in "CRZY". In RIT, CRZY shares on each market are listed with the tickers "CRZY_M" and "CRZY_A".

Background: Since deregulation in the late 1990's and early 2000's, North American stock exchanges have faced increased competition from start-up exchanges (often referred to as Electronic Communication Networks or ECNs). These ECNs allow individuals to trade stocks that are "listed" on the NYSE, NASDAQ (or other exchanges). The ECNs compete by offering differentiated services, such as lower trading commissions, different/special order types, and extended trading hours. As of writing, there were over 40 different marketplaces in the United States where one can trade the same NYSE listed share of General Electric.

Regulatory: Most countries follow a methodology typically labeled "best execution". The concept of "best execution" is that a broker/individual/institution must always look for the best available price. While simple in practice, this becomes much more complex for a magnitude of reasons that we won't explore with this case. Best execution in relation to multiple marketplaces typically means that two markets can never be "crossed". A "crossed" market is one where its bid price exceeds the ask price of another market, or the ask price is less than the bid of another market. The following is an example:

| | Bid | Ask |
|---|---|---|
| **CRZY_M** | 10.02 | 10.05 |
| **CRZY_A** | 10.07 | 10.10 |

In this situation, someone has bid 10.07 for CRZY shares on the Alternate exchange, despite the fact that someone is willing to sell shares of CRZY on the Main exchange for $10.05. Submitting this bid

*Kevin Mak* and Tom McCurdy** prepared this case for the RIT market simulation platform, http://rit.rotman.utoronto.ca/.*
**Manager of the Financial Research and Trading Lab, Rotman School of Management;*
***Professor of Finance and Founding Director of the FRTL, Rotman School of Management, University of Toronto.*

is not logical (and not providing best execution) because they could buy the shares on the main exchange for $10.05[1]. They have just caused the market to be "crossed".

Submitting an order that will cause markets to be crossed is a regulatory breach because it violates best execution. However, the situation does still occur from time to time. Further, there are other scenarios (such as trading the same stock on different exchanges, in different countries) where there are no regulations against markets becoming crossed.

For the sake of this trading case, we relax the rule preventing crossed markets and we will develop an arbitrage algorithm to exploit these crossed markets by selling shares in the (relatively) more expensive market and buying shares in the cheaper market. In the example above, absent trading costs, one can buy shares of CRZY on the main exchange for $10.05 and immediately sell them on the alternate exchange for $10.07 and produce a 2 cent (per share) profit.

## Algorithmic Trading Simulation #1 – ALGO1

In the ALGO1 case, there will be one stock (CRZY) traded on two different exchanges (Main and Alternate). The two stocks will closely mirror each other, with the bids and offers of the two different markets becoming occasionally crossed. Write a trading algorithm that will identify situations when the market is crossed and automatically submit trades to profit from the scenario.

There is a limit of 10,000 shares per order, and each student is limited to a position of 25,000 shares gross or net. There are no trading commissions charged on either exchange. Trading will take place over 5 minutes (300 seconds). For the ALGO1 case, the Rotman Application Programming Interface (API) is enabled and can be accessed in Microsoft Excel in the following manner:

1. Turn on the Developer ribbon
2. In the Visual Basic window, go to "Tools → References" and add "Rotman Interactive Trader"
3. In any function or subroutine, use the following two lines of code to initialize the API:

```
Dim API As RIT2.API
Set API = New RIT2.API
```

4. A full list of API commands can then be accessed by using "API.<*Insertcommand*>" in VBA – for example, "API.AddOrder()"

[1] The real-world analogy of this would be putting a classified listing (bidding) to purchase an X-Box for $800, even though the stores are currently selling them for (asking) $400. An arbitrager can purchase one from the store for $400 and sell it to the person in the classifieds for $800 and pocket the $400 difference.

Discussion Questions and Follow Up:

(1) Make your algorithm more robust by adding cells in your worksheet that allow you to input trading costs (per share). Edit your algorithm logic to include the trading costs when checking for arbitrage opportunities.

(2) How would one change the algorithm logic to include more exchanges?

(3) Suppose one of the two stock exchanges was very slow due to bad technology (takes two seconds to execute orders that you have sent to it). How would that affect your algorithm and what would you consider doing to counteract these issues?

(4) How do bid and ask sizes (and market depth) affect your algorithm. Assuming the 10,000 per order share limit did not exist, what would you do to make your algorithm more efficient at exploiting opportunities? How would this affect the riskiness of your algorithm?

(5) Suppose the alternative exchange server crashed (halted), and the quotes were still on your computer but order routing was failing. How does your algorithm currently behave in this situation?